hbbdata Documentation

Release 1.0

Argonne National Laboratory

May 15, 2020

CONTENTS

CHAPTER

ONE

INSTALLING HBBDATA

1.1 General requirements

Generally, to property run and validate the package you need to have Python 3.x installed with the following packages:

- numpy
- scipy
- nose

All three are very common packages and are available for a variety of operating systems. The instructions below provide specific instructions on how to install hbbdata on Ubuntu Linux 18.04, CentOS 7, and Windows 10.

1.2 Ubuntu 18.04 instructions

Download the hbbdata distribution and uncompress it into the directory you want it installed to. Open a terminal and navigate to the root `hbbdata/` directory.

Install the required packages

```
sudo apt-get install python3 python3-numpy python3-scipy python3-nose
```

Run the install validation tests

```
nosetests3
```

If successful, the result of the testing should be `OK`, with the terminal output looking like

```
.....................................
----------------------------------------------------------------------
Ran 41 tests in 3.796s

OK
```

Add the `hbbdata/` root directory to your `PYTHONPATH` environment variable, preferably in your `.bashrc` file or similar so this step does not need to be repeated each time you open a new terminal. For example, if the full path to hbbdata is `/path/to/hbbdata` and you are using the bash shell, add the following line to your `.bashrc`

```
export PYTHONPATH=$PYTHONPATH:/path/to/hbbdata
```

Now see the *Quickstart guide* for an example of how to use hbbdata.

1.3 CentOS 7 instructions

Download the hbbdata distribution and uncompress it into the directory you want it installed to. Open a terminal and navigate to the root `hbbdata/` directory.

Install the IUS set of packages, which contains a python 3 release

```
sudo yum install https://centos7.iuscommunity.org/ius-release.rpm
```

Install the required packages

```
sudo yum install python36 python36-numpy python36-scipy python36-nose
```

Run the install validation tests

```
nosetests-3.6
```

If successful, the result of the testing should be `OK`, with the terminal output looking like

```
.....................................
----------------------------------------------------------------------
Ran 41 tests in 3.796s

OK
```

Add the `hbbdata/` root directory to your `PYTHONPATH` environment variable, preferably in your `.bashrc` file or similar so this step does not need to be repeated each time you open a new terminal. For example, if the full path to hbbdata is `/path/to/hbbdata` and you are using the bash shell, add the following line to your `.bashrc`

```
export PYTHONPATH=$PYTHONPATH:/path/to/hbbdata
```

Now see the *Quickstart guide* for an example of how to use hbbdata.

1.4 Windows 10 instructions

These directions assume the use of the Python 3.7 Anaconda distribution to run Python on Windows. Other methods, including a bare install from the official Python 3.7 binaries are possible, but Anaconda provides the quickest method for installing the dependencies. These directions assume Anaconda has been installed.

Unzip the hbbdata distribution into the directory you want it installed into.

Open an Anaconda terminal (generally click the Start Menu, then navigate to click on `Anaconda Prompt`). Navigate to the hbbdata root directory using the terminal, i.e.

```
cd C:/Users/username/hbbdata
```

assuming `C:/Users/username` is the directory in which you install the package.

Run the install validation tests (all the required packages are already installed in the base Anaconda installation).

```
nosetests
```

If successful, the result of the testing should be `OK`, with the terminal output looking like

```
.....................................
----------------------------------------------------------------------
Ran 41 tests in 3.796s

OK
```

Use `conda-develop` to add hbbdata to your python path. Identify the full pathname to the directory you installed hbbdata. Say this path is `C:/Users/username/hbbdata` then run the command

```
conda-develop C:/Users/username/hbbdata
```

Now see the *Quickstart guide* for an example of how to use hbbdata.

CHAPTER

TWO

QUICKSTART GUIDE

This guide assumes you have followed the installation instructions and have added hbbdata to your `PYTHONPATH`.

Open up a terminal and run Python 3.7. First test to make sure you can import the hbbdata module:

```
>>> import hbbdata
```

If this command runs without error keep going. If not go back to *Installing hbbdata*.

The hbbdata API is divided into a number of modules, each containing logically similar functions accessing the Division 5 design data. For more details see *API description and documentation*.

As an example, let's find the ASME Section II Young's modulus of 316SS at 700°C. Simply type:

```
>>> from hbbdata import elastic
>>> elastic.youngs("316", 700)
140000.0
```

Or we could look up the design yield strength of Alloy 800H at 550°C:

```
>>> from hbbdata import plastic
>>> plastic.yield_stress("800H", 550)
108.0
```

A slightly more complicated example might be to determine if the point represented by a fatigue damage fraction of 0.1 and a creep damage fraction of 0.4 falls within the design creep-fatigue interaction diagram from 2-1/4Cr-1Mo steel:

```
>>> from hbbdata import interaction
>>> interaction.inside_envelope("2.25Cr-1Mo", 0.1, 0.4)
False
```

As a final example, find the strain corresponding to a stress of 120 MPa for the 9Cr-1Mo-V isochronous stress-strain curve for 525°C and 120,000 hours design life:

```
>>> from hbbdata import isochronous
>>> isochronous.strain_to_time_stress("gr91", 525, 120000.0, 220)
0.05559105838726914
```

All of the design data can be accessed through these and similar functions, which are fully described in the *API description and documentation*.

CHAPTER

THREE

API DESCRIPTION AND DOCUMENTATION

3.1 General overview

This section describes the complete public API of the hbbdata package. In general, functions providing Section III, Division 5 design data will take as parameters a material, a temperature, other design parameters, and optional parameters describing options in the Code that are not always taken.

The convention for relating each Class A material to a string key, used in identifying the material in the package, is given in the following table:

Table 1: Material keys

Material	Key
304SS	304
316SS	316
Ni-Fe-Cr UNS N08810	800H
2-1/4Cr-1Mo	2.25Cr-1Mo
9Cr-1Mo-V	gr91
Alloy 617	A617

In all cases package unites are consistent with the following unit system:

Table 2: Unit system

Type	Units
Stress	MPa
Temperature	°C
Time	hours
Strain	mm/mm

3.2 API documentation

The hbbdata package is divided into a number of submodules, each containing logically related sets of design data functions. The following sections document each module individually.

In addition to this public API, the package also contains a number of helper functions designed to read and interpolate data and aid in testing. These functions are documented with Python docstrings in their individual source files.

3.2.1 elastic

This module returns the Code values of elastic properties as functions of temperature.

hbbdata.elastic.**poissons**(*material*)
The ASME Poissons ratios are temperature independent so this function just returns the relevant value.

Parameters **material** – the HBB class A material

hbbdata.elastic.**youngs**(*material*, *temp*, *extrapolate=False*)
The code specifies the Young's modulus as a function of temperature

Parameters

- **material** – the HBB class A material
- **temp** – the temperature(s) to return values for, C

3.2.2 thermal

This module return thermal properties (coefficient of thermal expansion, thermal conductivity, and thermal diffusivity) from Section II

hbbdata.thermal.**ctc**(*material*, *temp*, *extrapolate=False*)
The code specified instantaneous CTC as a function of temperature

Parameters

- **material** – the HBB class A material
- **temp** – the temperature(s) to return values for, C

Keyword Arguments **extrapolate** – if true extrapolate properties

hbbdata.thermal.**ctd**(*material*, *temp*, *extrapolate=False*)
The code specified instantaneous CTD as a function of temperature

Parameters

- **material** – the HBB class A material
- **temp** – the temperature(s) to return values for, C

Keyword Arguments **extrapolate** – if true extrapolate properties

hbbdata.thermal.**cte**(*material*, *temp*, *extrapolate=False*)
The code specified instantaneous CTE as a function of temperature

Parameters

- **material** – the HBB class A material
- **temp** – the temperature(s) to return values for, C

Keyword Arguments **extrapolate** – if true extrapolate properties

3.2.3 plastic

This module returns Code values of yield stress and ultimate tensile stress

hbbdata.plastic.**tensile_strength_reduction_factor**(*material*, *time*, *temp*)
Return the tensile strength reduction factor for the given material, time, and temperature.

Parameters

- **material** – the HBB class A material
- **time** – time (in hours) of service @ temperature
- **temperature** – service temperature, degrees C

hbbdata.plastic.**ultimate_tensile_stress**(*material*, *temp*, *reduce_temp=None*, *time=None*)
Code Su as a function of temperature

Parameters

- **material** – the HBB class A material
- **temperature** – the temperature(s) to return values for, C

Keyword Arguments

- **reduce_temp** – temperature to use for the strength reduction factor Default (None) means use the actual temperature
- **time** – service time or equivalent service time @ temperature, for applying strength reduction factor Default (None) implies no reduction

hbbdata.plastic.**yield_strength_reduction_factor**(*material*, *time*, *temp*)
Return the yield strength reduction factor for the given material, time, and temperature.

Parameters

- **material** – the HBB class A material
- **time** – time (in hours) of service @ temperature
- **temperature** – service temperature, degrees C

hbbdata.plastic.**yield_stress**(*material*, *temp*, *reduce_temp=None*, *time=None*)
Code Sy as a function of temperature

Parameters

- **material** – the HBB class A material
- **temperature** – the temperature(s) to return values for, C

Keyword Arguments

- **reduce_temp** – temperature to use for the strength reduction factor Default (None) means use the actual temperature
- **time** – service time or equivalent service time @ temperature, for applying strength reduction factor Default (None) implies no reduction

3.2.4 intensities

Functions returning the various primary stress allowable intensities

hbbdata.intensities.**S_m**(*material*, *temp*, *reduce_temp=None*, *time=None*, *RT=20.0*, *no_reduction=False*)

Code time-independent allowable S_m as a function of temperature

Parameters

- **material** – valid HBB Class A material
- **temp** – temperature requested

Keyword Arguments

- **reduce_temp** – temperature to use for the strength reduction factor. Default (None) means use the actual temperature
- **time** – time or equivalent time, for applying strength reduction factors Default (None) does not apply a reduction
- **RT** – the value of room temperature, for determining the allowable
- **no_reduction** – override the time/temperature factors and return unaged properties (Default False)

Source: Code rules, strictly in Section II. However a good summary is in HBB-2160 (3), (-b) to (-f)

S_m is the minimum of:

i) 1/3 * Su @ room temperature

ii) 1/3 * Su @ temperature

iii) 2/3 * Sy @ room tepmerature

iv) **For 304, 316, A800H, and A617:** 90% * Sy @ temperature

For the rest: 2/3 * Sy @ temperature

hbbdata.intensities.**S_mt**(*material*, *temp*, *time*, *extrapolate=False*, *reduce_temp=None*, *effective_time=None*, *RT=20.0*, *no_reduction=False*)

Code S_mt allowable stress intensity as a function of temperature and time

Parameters

- **material** – HBB Class A material
- **temp** – temperature, in degrees C
- **time** – time, hours

Keyword Arguments

- **extrapolate** – extrapolate off the Code table, applies only to S_t
- **reduce_temp** – temperature to use for time-temperature reduction factor, defaults to actual temperature
- **effective_time** – time for time-temperature reduction factor, defaults to actual time
- **RT** – room temperature for that part of the Sm definition
- **no_reduction** – override the time/temperature factors and return unaged properties

hbbdata.intensities.**S_o**(*material*, *temp*)
Code design allowable stress intensity S_o as a function of temperature

Parameters

- **material** – HBB Class A material
- **temp** – temperature, in degrees C

hbbdata.intensities.**S_t**(*material*, *temp*, *time*, *extrapolate=False*)
Code time-dependent allowable stress intensity S_t as a function of temperature and time

Parameters

- **material** – valid HBB Class A material
- **temp** – temperature, degree C
- **time** – time, hours

Keyword Arguments **extrapolate** – extrapolate outside of Code tables

hbbdata.intensities.**time_S_t**(*material*, *temp*, *stress*, *extrapolate=True*)
Return Code value of time to a given S_t

Parameters

- **material** – the HBB Class A material
- **temp** – temperature
- **stress** – S_t

Keyword Arguments

- **extrapolate** – extrapolate outside Code table
- **ztol** – tolerance for checking if function value is flat
- **bigmax** – big bracket value for brentq

3.2.5 rupture

Module returns rupture stresses and times from Section III, Division 5

hbbdata.rupture.**S_r**(*material*, *temp*, *time*, *extrapolate=False*)
Returns the Code Sr value

Parameters

- **material** – the HBB Class A material
- **temp** – the temperature to get data for, C
- **time** – the time to get data for, hrs

Keyword Arguments **extrapolate** – extrapolate out of the table

hbbdata.rupture.**time_rupture**(*material*, *temp*, *stress*, *extrapolate=True*)
Returns the Code value of time to rupture

Parameters

- **material** – the HBB Class A material
- **temp** – the temperature to get data for, C
- **stress** – the rupture stress, MPa

Keyword Arguments

- **extrapolate** – if true, extrapolate out of the table
- **ztol** – tolerance for checking if the function value is flat
- **bigmax** – maximum time for extrapolation

3.2.6 fatigue

Functions returning Code fatigue curve data and generated related data.

hbbdata.fatigue.**cycles_to_failure**(*material*, *maxtemp*, *erange*, *extrapolate=True*, *cycles_min=0.0*, *cycles_max=1000000000000.0*)

Number of cycles to failure for a given strain range

Parameters

- **material** – HBB Class A material
- **maxtemp** – maximum cycle temperature
- **erange** – strain range

Keyword Arguments

- **extrapolate** – extrapolate outside the table (Default=True)
- **cycles_min** – minimum number of cycles, for Brent's method (Default=0.0)
- **cycles_max** – maximum number of cycles, for Brent's method (Default=1.0e12)

hbbdata.fatigue.**strain_to_failure**(*material*, *maxtemp*, *cycles*, *extrapolate=True*)

Strain to failure for a given number of cycles

Parameters

- **material** – HBB Class A material
- **maxtemp** – maximum cycle temperature
- **cycles** – number of cycles

Keyword Arguments **extrapolate** – extrapolate outside the table (Default=True)

3.2.7 interaction

This module handles checks of the Code creep-fatigue interaction diagrams

hbbdata.interaction.**distance_envelope**(*material*, *damage_fatigue*, *damage_creep*)

Distance from a point to the envelope, sign is positive if inside the envelop and negative if outside

Parameters

- **material** – HBB Class A material
- **damage_fatigue** – fatigue damage fraction
- **damage_creep** – creep damage fraction

hbbdata.interaction.**inside_envelope**(*material*, *damage_fatigue*, *damage_creep*)

Return True if the point lies in the design envelope and False if not

Parameters

- **material** – HBB Class A material
- **damage_fatigue** – fatigue damage fraction
- **damage_creep** – creep damage fraction

hbbdata.interaction.**interaction_fatigue**(*material*, *damage_fatigue*)

Enter the interaction diagram with a fatigue damage fraction and return the allowable creep damage fraction

Parameters

- **material** – HBB Class A material
- **damage_fatigue** – fatigue damage fraction

hbbdata.interaction.**interaction_creep**(*material*, *damage_creep*)

Enter the interaction diagram with a creep damage fraction and return the allowable fatigue damage fraction

Parameters

- **material** – HBB Class A material
- **damage_creep** – creep damage fraction

3.2.8 isochronous

This module returns values for isochronous and hot tensile curves for the Code materials.

hbbdata.isochronous.**isochronous**(*material*, *temperature*, *life*, *strain=array([0.*, *0.00044898*, *0.00089796*, *0.00134694*, *0.00179592*, *0.0022449*, *0.00269388*, *0.00314286*, *0.00359184*, *0.00404082*, *0.0044898*, *0.00493878*, *0.00538776*, *0.00583673*, *0.00628571*, *0.00673469*, *0.00718367*, *0.00763265*, *0.00808163*, *0.00853061*, *0.00897959*, *0.00942857*, *0.00987755*, *0.01032653*, *0.01077551*, *0.01122449*, *0.01167347*, *0.01212245*, *0.01257143*, *0.01302041*, *0.01346939*, *0.01391837*, *0.01436735*, *0.01481633*, *0.01526531*, *0.01571429*, *0.01616327*, *0.01661224*, *0.01706122*, *0.0175102*, *0.01795918*, *0.01840816*, *0.01885714*, *0.01930612*, *0.0197551*, *0.02020408*, *0.02065306*, *0.02110204*, *0.02155102*, *0.022]*))

Supply the isochronous curve at the provided temperature and design life.

Parameters

- **material** – HBB class A material
- **temperature** – temperature, in C
- **life** – design life, in hours

Keyword Arguments **strain** – nominal engineering strains to return

hbbdata.isochronous.**hot_tensile**(*material*, *temperature*, *strain=array([0.*, *0.00044898*, *0.00089796*, *0.00134694*, *0.00179592*, *0.0022449*, *0.00269388*, *0.00314286*, *0.00359184*, *0.00404082*, *0.0044898*, *0.00493878*, *0.00538776*, *0.00583673*, *0.00628571*, *0.00673469*, *0.00718367*, *0.00763265*, *0.00808163*, *0.00853061*, *0.00897959*, *0.00942857*, *0.00987755*, *0.01032653*, *0.01077551*, *0.01122449*, *0.01167347*, *0.01212245*, *0.01257143*, *0.01302041*, *0.01346939*, *0.01391837*, *0.01436735*, *0.01481633*, *0.01526531*, *0.01571429*, *0.01616327*, *0.01661224*, *0.01706122*, *0.0175102*, *0.01795918*, *0.01840816*, *0.01885714*, *0.01930612*, *0.0197551*, *0.02020408*, *0.02065306*, *0.02110204*, *0.02155102*, *0.022])*)

Supply the hot tensile curve at the requested temperature

Parameters

- **material** – HBB class A material
- **temperature** – temperature, in C

Keyword Arguments **strain** – nominal engineering strains to return

hbbdata.isochronous.**strain_to_time_stress**(*material*, *temperature*, *time*, *stress*)

Calculate the strain at a given temperature, time and stress

Parameters

- **material** – HBB class A material
- **temperature** – temperature, in C
- **time** – time, in hours
- **stress** – stress, in MPa

hbbdata.isochronous.**issc_relaxation_analysis_stress**(*material*, *temperature*, *stress*, *times*)

Perform a code method of isochronous curves relaxation analysis hitting each of the indicated times.

This routine is stress-based

Parameters

- **material** – HBB class A material
- **temperature** – temperature, in C
- **stress** – initial stress, in MPa
- **times** – times to solve for, in hours

hbbdata.isochronous.**issc_relaxation_analysis_strain**(*material*, *temperature*, *strain*, *times*)

Perform a code method of isochronous curves relaxation analysis hitting each of the indicated times.

This routine is strain-based

Parameters

- **material** – HBB class A material
- **temperature** – temperature, in C
- **strain** – initial strain, mm/mm
- **times** – times to solve for, in hours

3.2.9 epp

This module contains tools for the Section III, Division 5 Elastic Perfectly Plastic (EPP) Code cases.

hbbdata.epp.**pseudoyield_N861**(*material*, *life*, *target_strain*, *temp=array([40., 51.42857143, 62.85714286, 74.28571429, 85.71428571, 97.14285714, 108.57142857, 120., 131.42857143, 142.85714286, 154.28571429, 165.71428571, 177.14285714, 188.57142857, 200., 211.42857143, 222.85714286, 234.28571429, 245.71428571, 257.14285714, 268.57142857, 280., 291.42857143, 302.85714286, 314.28571429, 325.71428571, 337.14285714, 348.57142857, 360., 371.42857143, 382.85714286, 394.28571429, 405.71428571, 417.14285714, 428.57142857, 440., 451.42857143, 462.85714286, 474.28571429, 485.71428571, 497.14285714, 508.57142857, 520., 531.42857143, 542.85714286, 554.28571429, 565.71428571, 577.14285714, 588.57142857, 600.])*, *disable_yield=False*)

The EPP strain limits pseudoyield stress as a function of temperature

Parameters

- **material** – HBB class A material
- **life** – design life, in hours. A life of 0 is interpreted to mean use the hot tensile curve
- **target_strain** – the target strain

Keyword Arguments

- **temp** – temperature(s) to return
- **disable_yield** – if true skip the yield check

hbbdata.epp.**pseudoyield_N862**(*material*, *t_prime*, *temp=array([20., 31.83673469, 43.67346939, 55.51020408, 67.34693878, 79.18367347, 91.02040816, 102.85714286, 114.69387755, 126.53061224, 138.36734694, 150.20408163, 162.04081633, 173.87755102, 185.71428571, 197.55102041, 209.3877551, 221.2244898, 233.06122449, 244.89795918, 256.73469388, 268.57142857, 280.40816327, 292.24489796, 304.08163265, 315.91836735, 327.75510204, 339.59183673, 351.42857143, 363.26530612, 375.10204082, 386.93877551, 398.7755102, 410.6122449, 422.44897959, 434.28571429, 446.12244898, 457.95918367, 469.79591837, 481.63265306, 493.46938776, 505.30612245, 517.14285714, 528.97959184, 540.81632653, 552.65306122, 564.48979592, 576.32653061, 588.16326531, 600.])*)

The pseudoyield stress used in Code Case N-862

Parameters

- **material** – HBB class A material
- **t_prime** – trial life

Keyword Arguments **temp** – temperatures requested

3.2.10 limits

This module implements various temperature limits used in the Code

hbbdata.limits.**Sm_St**(*material*, *time=100000.0*, *ll=425.0*, *ul=600.0*)
The S_m == S_t for 100,000 hours criteria used in HBB-T

Parameters **material** – the HBB class A material

Keyword Arguments

- **time** – the time to use in the equality, default 100,000 hours
- **ll** – lower limit to use in solving for the critical point
- **ul** – upper limit to use in solving for the critical point

hbbdata.limits.**Sr_max_T**(*material*)
The hot temperature as defined by HAA-1330-1

Parameters **material** – the HBB class A material

hbbdata.limits.**T_max**(*material*)
The hot temperature as defined by HAA-1330-1

Parameters **material** – the HBB class A material

CHAPTER

FOUR

SOFTWARE QUALITY PLAN

4.1 Software development plan

This chapter describes the software development quality plan used to create the hbbdata package. This particular chapter focuses on development. Details on *Verification and Validation* are contained in a separate chapter.

4.1.1 Requirements

The main purpose of this package is to provide an API to access the design information contained in Section III, Division 5, Subsection HB, Subpart B of the ASME Boiler & Pressure Vessel Code along with appropriate ancillary information from Section II, Part D and relevant Nuclear Code Cases.

Subpart HBB contains rules for the design of Class A elevated temperature nuclear components. The Subpart allows the use of a limited selection of materials, listed below. In addition, a recently passed Nuclear Code Case allows the use of Alloy 617 for Class A construction. This package includes the Alloy 617 design data, drawn from the current version of the Code Case. The materials covered by the package are then

- 304SS (304) – 304H austenitic stainless steel
- 316SS (316) – 316H austenitic stainless steel
- Ni-Fe-Cr UNS N08810 (800H) – Alloy 800H Ni-alloy
- 2-1/4Cr-1Mo (2.25Cr-1Mo) – annealed 2-1/4Cr-1Mo steel
- 9Cr-1Mo-V (gr91) – modified 9Cr-1Mo, Grade 91 ferritic-martensitic steel
- Alloy 617 (A617) – 52Ni-22Cr-13Co-9Mo, Alloy 617 (UNS N06617) Ni-alloy

The string keys in parenthesis are important – these short descriptors are used by the package API to refer to each material.

For a given combination of material and design parameters (almost always temperature and often design life) the package will return the requested material property. If the combination of requested property, material, and design parameters is invalid, for example the requested temperature is not in the Subpart HBB design range, then the package should return an error.

The Code provides the design data, for the most part, as tables in either Section III, Division 5, Subsection HB, Subpart B, Section II, or the Alloy 617 Code Case. In general, designers will need the design information at points between the tabulated entries. The Code does not specify a particular method of interpolation. The requirement specification here selects a method of interpolation based on engineering judgement, always either linear or log-linear interpolation. For the tabulated design data the package must correctly read in the Code table defining the property, check the requested design parameters for validity and through an error, if required, and then correctly interpolate the table to the requested value of the parameters.

There are two exceptions to this general requirement. One is that the isochronous stress-strain curves are provided as figures in the Code, without corresponding tabulated data. The isochronous curves provide strain as a function of the design parameters of temperature, stress, and time. To provide arbitrary values of the isochronous stress-strain relation the package implements the original deformation models underlying the isochronous stress-strain curves. This strategy is outlined in the API reference on the curves. However, the intent remains the same – the package must correctly provide values of the curves for arbitrary sets of design parameters.

The second exception are design information based in turn on other design data. For example, the package must provide minimum time-to-rupture for a given temperature and stress, whereas the Code provides minimum stress-to-rupture for given temperatures and times. For this case, the required design information is a mathematical rearrangement of other design data and so the package is required to accurately implement the corresponding mathematical formula.

The following sections describe the design data the package must implement. Each section follows a common format. It starts with a general overview of the design data, including the units provided by the package. It then lists the location(s) in the Code providing the information. Then the section describes the input design parameter and any limits on those design parameters. Finally, the section lists the method of implementation – either reading and interpolating from a table (listing the interpolation method) or some mathematical relation to existing design data.

Young's modulus

Description and units

The elastic Young's modulus of the material in MPa.

Code reference

Section II, Part D, Tables TM-1 through TM-5.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperature in the corresponding table in Section II, Part D.

Implementation type

Tabulated values with linear interpolation in temperature.

Poisson's ratio

Description and units

The elastic Poisson's ratio of the material, unitless.

Code reference

Section II, Part D, Table PRD.

Parameters and limits

None, ASME values are constant in temperature.

Implementation type

Single value.

Thermal expansion coefficient

Description and units

The instantaneous thermal expansion coefficient, units of mm/mm/°C.

Code reference

Section II, Part D, Tables TE-1 through TE-5.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperature in the corresponding table in Section II, Part D.

Implementation type

Tabulated values with linear interpolation in temperature.

Thermal conductivity

Description and units

Thermal conductivity, units of W/(m °C).

Code reference

Section II, Part D, Table TCD.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperature in the corresponding table in Section II, Part D.

Implementation type

Tabulated values with linear interpolation in temperature.

Thermal diffusivity

Description and units

Thermal diffusivity, units of 10^{-6} m^2/s.

Code reference

Section II, Part D, Table TCD.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperature in the corresponding table in Section II, Part D.

Implementation type

Tabulated values with linear interpolation in temperature.

Yield strength

Description and units

Code value of yield strength (S_y), units of MPa. Optionally reduced for the effect of time and temperature per HBB-2160.

Code reference

Section II, Part D, Subpart 1, Table Y-1 extended in Section III, Division 5, HBB-I-14.5.

Parameters and limits

Temperature (°C), limited to the minimum temperature in Section II, Table Y-1 and the maximum temperature in HBB-I-14.5.

Implementation type

Tabulated values with linear interpolation in temperature.

Tensile strength

Description and units

Code value of tensile strength (S_u), units of MPa. Optionally reduced for the effect of time and temperature per HBB-2160.

Code reference

Section II, Part D, Subpart 1, Table U extended in Section III, Division 5, HBB-3225-1.

Parameters and limits

Temperature (°C), limited to the minimum temperature in Section II, Table U and the maximum temperature in HBB-3225-1.

Implementation type

Tabulated values with linear interpolation in temperature.

Yield strength reduction factor

Description and units

Time-temperature reduction factor for the yield strength, unitless.

Code reference

Section III, Division 5, Subsection HB, Subpart B Tables HBB-3225-2 and HBB-3225-3A.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB tables, and time (hours), limited to be between zero and the maximum design life for the material given by the Subpart HBB allowable stress tables.

Implementation type

Tabulated values with linear interpolation in temperature and linear interpolation in time or fixed values (depending on the material).

Tensile strength reduction factor

Description and units

Time-temperature reduction factor for the tensile strength, unitless.

Code reference

Section III, Division 5, Subsection HB, Subpart B Tables HBB-3225-2, HBB-3225-3B, and HBB-3225-4.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB tables, and time (hours), limited to be between zero and the maximum design life for the material given by the Subpart HBB allowable stress tables.

Implementation type

Tabulated values with linear interpolation in temperature and linear interpolation in time or fixed values (depending on the material).

Allowable stress S_m

Description and units

Section III, Division 5 allowable stress S_m, in MPa, optionally reduced for time-temperature effects per HBB-2160.

Code reference

HBB-2160(3),(-b) to (-f)

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB tables for yield and tensile stress.

Implementation type

A mathematical relation to the Code values of yield and tensile strength, described in the above-cited section of the Code.

Allowable stress S_t

Description and units

Section III, Division 5 allowable stress S_t, in MPa.

Code reference

Tables HBB-I-14.4(A to E).

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table, and time (hours), limited to the maximum value given in the corresponding HBB table.

Implementation type

Tabulated, interpolated linearly in temperature and log-linearly in time.

Allowable stress S_{mt}

Description and units

Section III, Division 5 allowable stress S_{mt}, in MPa.

Code reference

Defined by HBB-3221 as the lesser of S_m and S_t, where S_m is optionally reduced for time-temperature effects per HBB-2160.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table, and time (hours), limited to the maximum value given in the corresponding HBB table.

Implementation type

Mathematical relation between S_m and S_t.

Allowable stress S_o

Description and units

Section III, Division 5 allowable stress S_o, in MPa.

Code reference

Table HBB-I-14.2.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table.

Implementation type

Tabulated, interpolated linearly in temperature.

Time to indicated S_t

Description and units

Given an allowable stress S_t and a temperature return the time to that allowable stress.

Code reference

Tables HBB-I-14.4(A to E).

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table, and stress (MPa), limited to the tabulated values in that table.

Implementation type

Mathematical rearrangement of the S_t table.

Minimum stress to rupture S_r

Description and units

Section III, Division 5 minimum stress to rupture S_r, in MPa.

Code reference

Table HBB-I-14.6(A to E)

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table, and time (hours), limited to the maximum time in the corresponding HBB table.

Implementation type

Tabulated, interpolated linearly in temperature and log-linearly in time.

Time to indicated S_r

Description and units

Given an allowable stress S_r and a temperature return the time in hours to that rupture stress.

Code reference

Table HBB-I-14.6(A to E)

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperatures in the corresponding HBB table, and stress (MPa), limited to the tabulated values in that table.

Implementation type

Mathematical rearrangement of the S_r table.

Strain range to allowable design cycles ε_t

Description and units

Strain range (mm/mm) giving the provided cycles to failure.

Code reference

Figures HBB-T-1410-1(A to E)

Parameters and limits

Temperature (°C), limited to the maximum temperature for which there is a fatigue curve in HBB-T-1410, and number of design cycles, limited to the maximum values in HBB-T-1410.

Implementation type

Tabulated. Bounding in temperature – use fatigue curve closest, but greater, than the provided temperature. Interpolate log-linearly in cycles.

Allowable design cycles N_d

Description and units

Design allowable fatigue cycles.

Code reference

Figures HBB-T-1410-1(A to E)

Parameters and limits

Temperature (°C), limited to the maximum temperature for which there is a fatigue curve in HBB-T-1410, and the strain range (mm/mm).

Implementation type

Mathematical rearrangement of the design fatigue curves.

Creep-fatigue interaction diagram

Description and units

Report whether the provided fatigue damage (D_f) and creep damage (D_c) falls within the Code design creep-fatigue interaction diagram.

Code reference

Figure HBB-T-1410-2.

Parameters and limits

Unitless input creep and fatigue damage fractions.

Implementation type

Fully described by Figure HBB-T-1410-2.

Isochronous stress-strain curve values

Description and units

Strains (mm/mm) corresponding to the provided design parameters describing the total deformation accumulated by the material under those conditions.

Code reference

Figures in HBB-T-1800.

Parameters and limits

Temperature (°C), limited to the maximum and minimum temperatures for which HBB-T-1800 provides a design curve. Stress, in MPa. Time, in hours, limited to the maximum time curve provided in HBB-T-1800.

Implementation type

Unique among all the design data provided by the package. The Code provides the plotted isochronous curves in HBB-T-1800. However, it is very difficult to provide arbitrary values of strain based on these figures. Instead, the package should implement the deformation models underlying the Code isochronous curves directly. These equations generally match the Code plots. Small discrepancies will be corrected by future Code action.

Hot tensile curve values

Description and units

Strains (mm/mm) corresponding to the provided design parameters describing the time-independent deformation accumulated by the material under those conditions.

Code reference

Figures HBB-T-1800.

Parameters and limits

Temperature (°C), limited to the maximum temperature for which there is a design curve given in HBB-T-1800.

Implementation type

The zero-time isochronous stress-strain curve.

Pseudo yield stress for Code Case N-861

Description and units

Pseudoyield stress (in MPa) for Code Case N-861 (EPP strain limits code case). Defined as the minimum of the Code yield strength at the provided temperature and the inelastic strain implied by the isochronous stress-strain curve for the given temperature, target strain, and time.

Code reference

Code Case N-861.

Parameters and limits

Temperature (°C), limited to the minimum and maximum temperature for which Section III, Division 5 provides an ischronous stress-strain curve. Design life (hours), limited to the maximum time for which Section III, Division 5 provides an isochronous curve. Target strain (mm/mm), limited to be less than 2%.

Implementation type

Mathematical rearrangement of the isochronous stress-strain curve values and Code yield strength.

Pseudo yield stress for Code Case N-862

Description and units

Pseudoyield stress (in MPa) for Code Case N-862 (EPP creep-fatigue code case). Defined as the minimum of the Code yield strength at the provided temperature and the Code minimum stress to rupture at the given temperature and time.

Code reference

Code Case N-861.

Parameters and limits

Temperature (°C) and time (hours), limited to the values for which there is a minimum stress to rupture (S_r) in Section III, Division 5.

Implementation type

Mathematical rearrangement of the minimum stress to rupture values and Code yield strength.

4.1.2 Configuration management

Package software development is managed on the internal Argonne National Laboratory GitLab system. This development system combines version control through a git repository, a ticketing system for requesting and documenting software development and changes, and access control to enforce quality control procedures.

Each feature, fulfilling one of the requirements above, will be developed on a separate branch in the git repository. Once the new branch completely implements the required feature it will be merged into the main code using a gitlab pull request. To accept the full request the following process should be followed:

1. Developers may not accept their own pull request. Instead a separate peer-review should be completed by another developer. This peer review should ensure 1) the pull request meets all the requirements listed here and 2) the code contained in the pull request is off good general quality, is clear, and well-documented with comments in the code.
2. In addition to the code, a pull request must provide a test suite fulfilling the requirements in *Verification and Validation*. The composite test suite, once all the software features have been implemented, will serve as the final verification and validation method for the project.
3. The branch must pass the project python style requirements, as checked by the pylint tool.

Gitlab automatically enforces all three requirements – it will not allow a pull request to be merged unless it has undergone peer review, automatically passes all the tests, and meets the pylint style requirements. The peer-reviewer is responsible for ensuring the code is clear and documented with comments and that the developer has added sufficient tests to verify and validate the new features.

4.2 Software risk management

The primary risk is that the package will return inaccurate or invalid design data to a user request. The :ref`validation` plan is the primary method for mitigating this risk.

A secondary risk is a change to the end user's system configuration preventing the package from installing or running correctly. This risk is mitigated by writing the package in a cross-platform, interpreted language (Python) and providing installation checks to verify that the package is correctly installed on a new system.

CHAPTER

FIVE

VERIFICATION AND VALIDATION

Verification is the process of ensuring the software package meets the system requirements. Validation is ultimate acceptance by the end user. For the hbbdata package verification testing is accomplished through a two step approach:

1. For digitization of Code table or figure data, one developer impalements the digitization while a second developer peer-reviews the work by checking each entry against the corresponding table in the Code.
2. A unit test suite:
    1. Spot checks individual, interpolated values against randomly selected corresponding checks to the relevant Code book.
    2. Ensure that the appropriate out of bounds exceptions are thrown if the user requests data at conditions outside those allowed by the Code.

As this is a data package, our expectation is that the end user requirements are essentially accurate results, as checked with the verification procedure. As such, this part of the documentation focuses on the verification process. End user feedback on the API and python interface will be incorporated into the final, release version of the software.

5.1 Verification cross-checks

The tabulated Code data was entered into a digital format in the `data/` directory. The format used for 1D and 2D tables is such that the digitized versions of the table correspond, as much as possible, the actual Code book tables. This simplifies the process of cross-checking results. Each individual table was entered by the developer working on the related feature fulfilling a requirement in *Requirements*. A second, independent developer then cross-checked the table entry manually against the Code. This peer review process was documented in comments directly in the tables in `data/`.

5.2 Unit testing

As an additional layer of verification testing, the package contains python unit tests in the `test/` directory, which can be automatically run with the nose test harness. These unit tests do spot checks on the accuracy of the design data and also check to make sure the package will not return design data for invalid input parameters. As described in the *Installing hbbdata* section, these tests should be run by the user after installing the package to validate the integrity of the installation.

5.3 Isochronous curves

The isochronous stress-strain curves are the exception to this validation plan. As noted in the *Software quality plan* the strategy for implementing the isochronous stress-strain curves is *not* to digitize the Section III, Division 5 plots, but rather to implement the original deformation models underlying the plotted curves. This ensure that values can be easily calculated for arbitrary values of stress, time, and temperature. However, the process of uncovering and implementing the underlying models uncovered several small discrepancies between the current Division 5 isochronous curves and the original models. The following figures compare the original Division 5 curves (black lines) with the curves implemented in hbbdata (red lines). The differences are minor and would not significantly affect the design of a component. A Code change will change the Division 5 curves to match the model (red) curves shown here in the 2021 edition.

The figures cover the base Code materials (2.25Cr-1Mo, 9Cr-1Mo-V, Alloy 800H, 316SS, and 304SS). The implemented isochronous curves for Alloy 617 exactly match the design curves in the A617 Code Case.

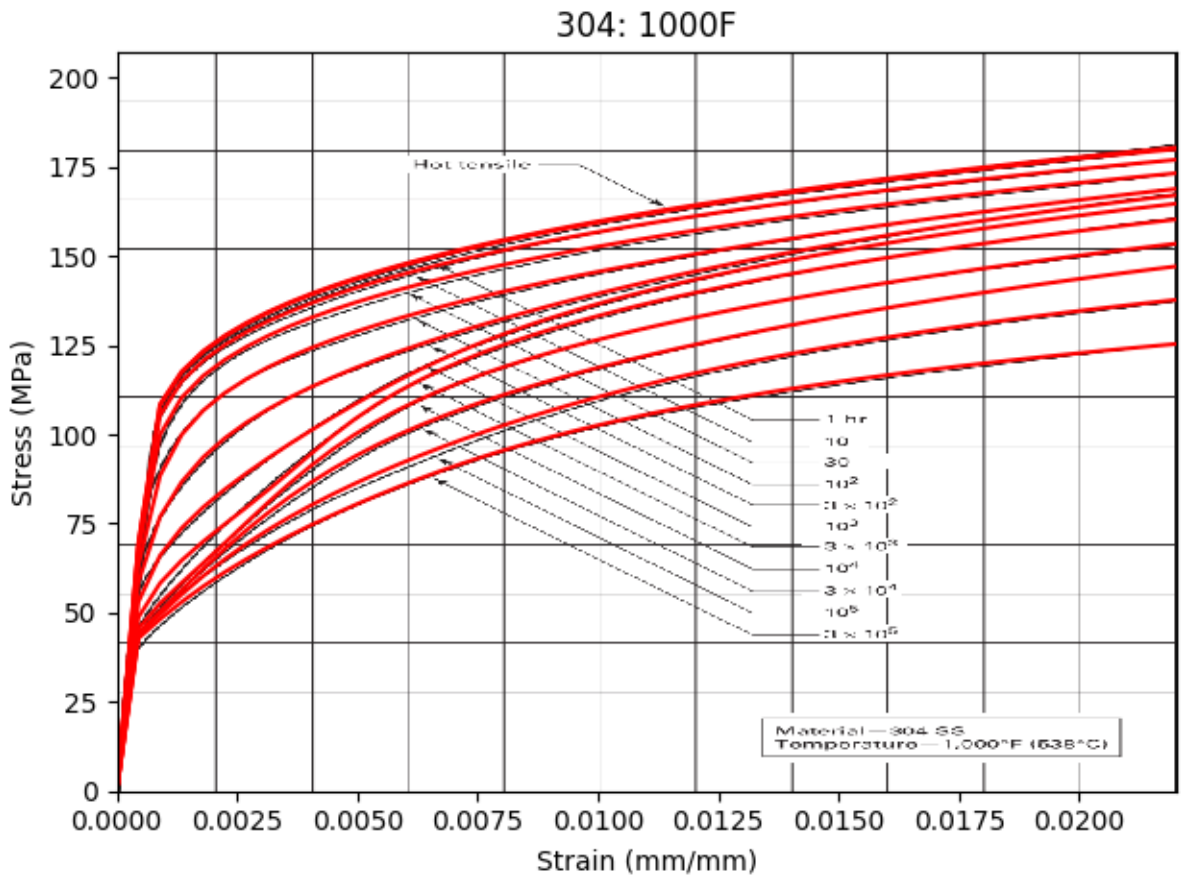

Fig. 1: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1000°F.

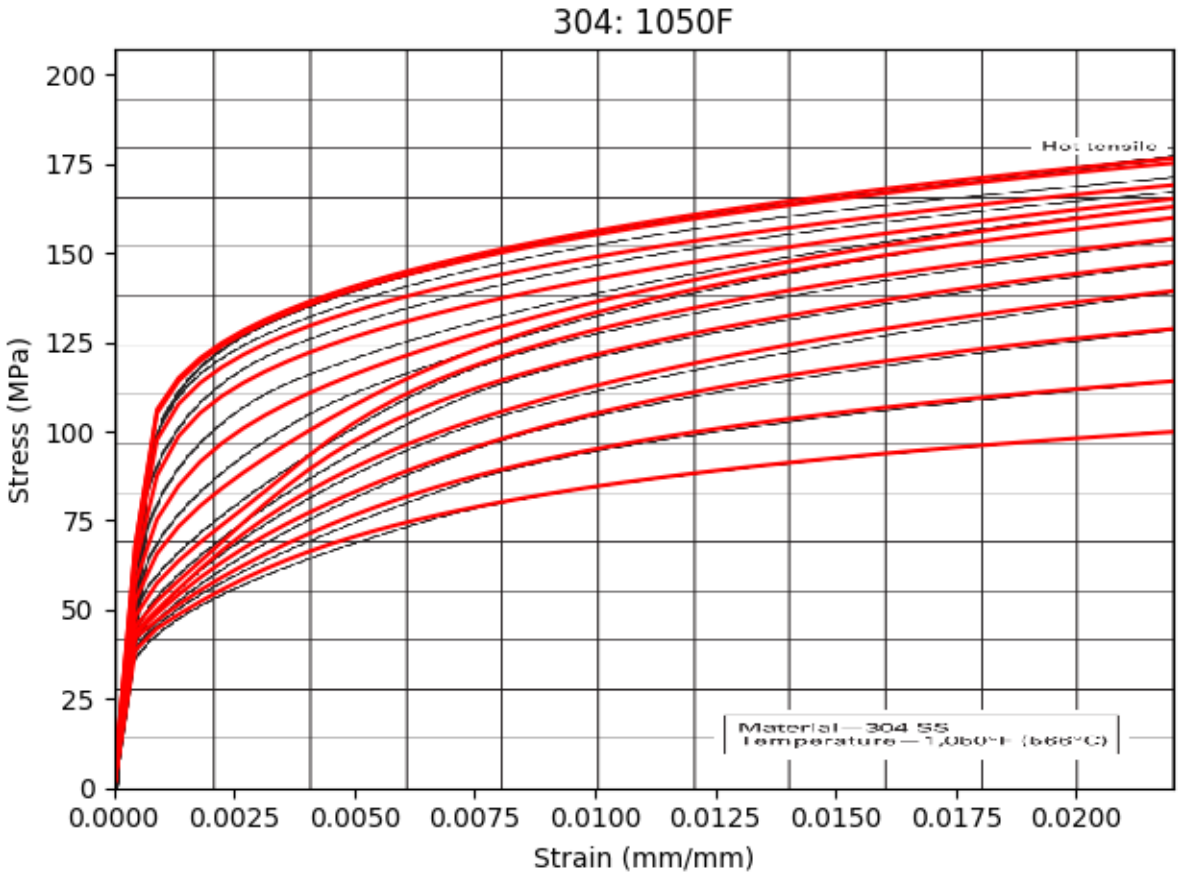

Fig. 2: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1050°F.

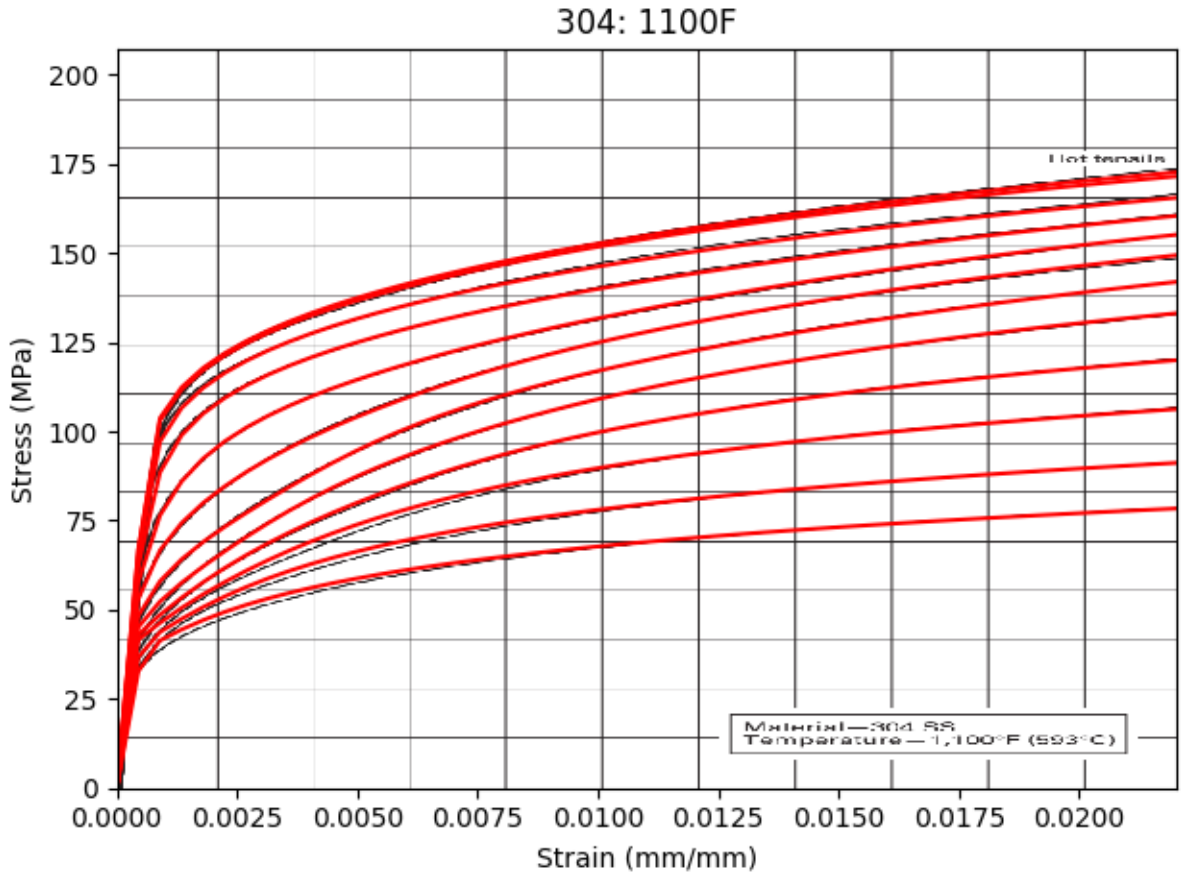

Fig. 3: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1100°F.

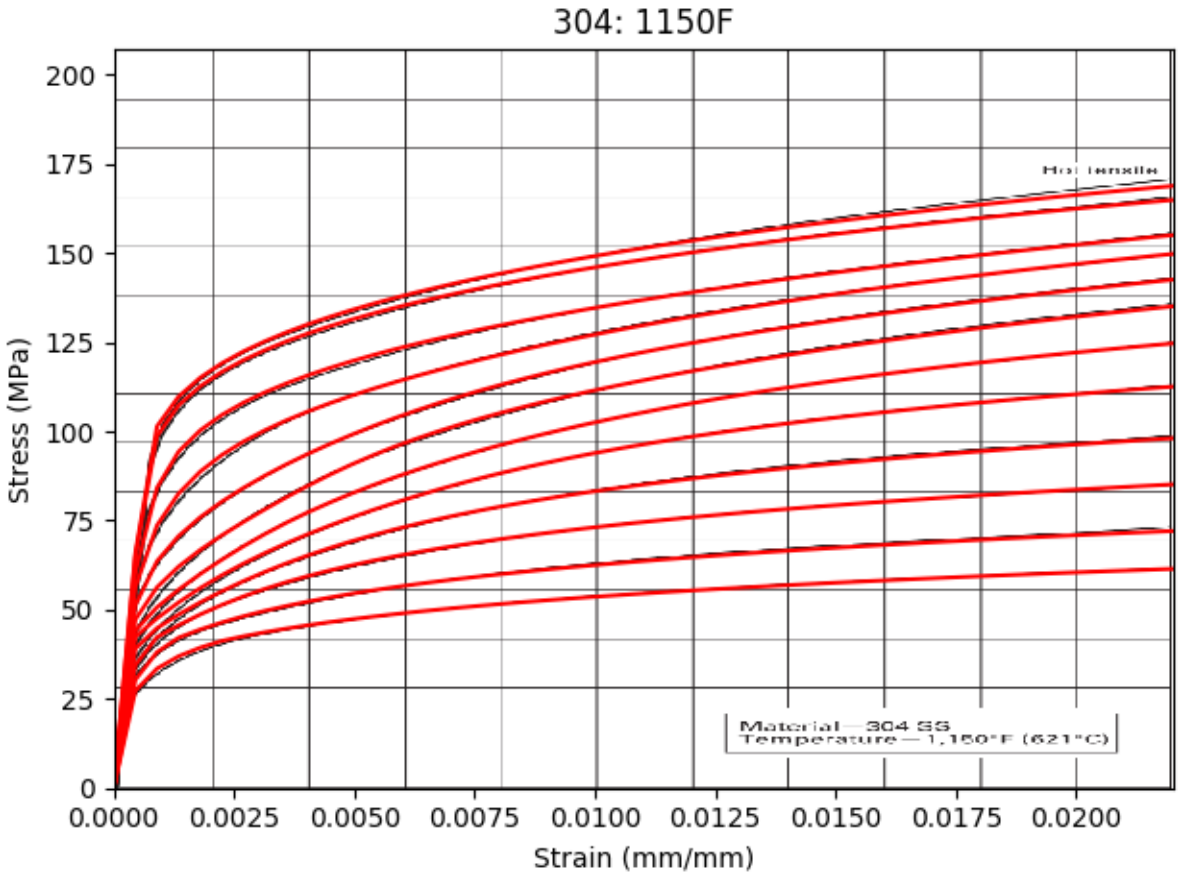

Fig. 4: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1150°F.

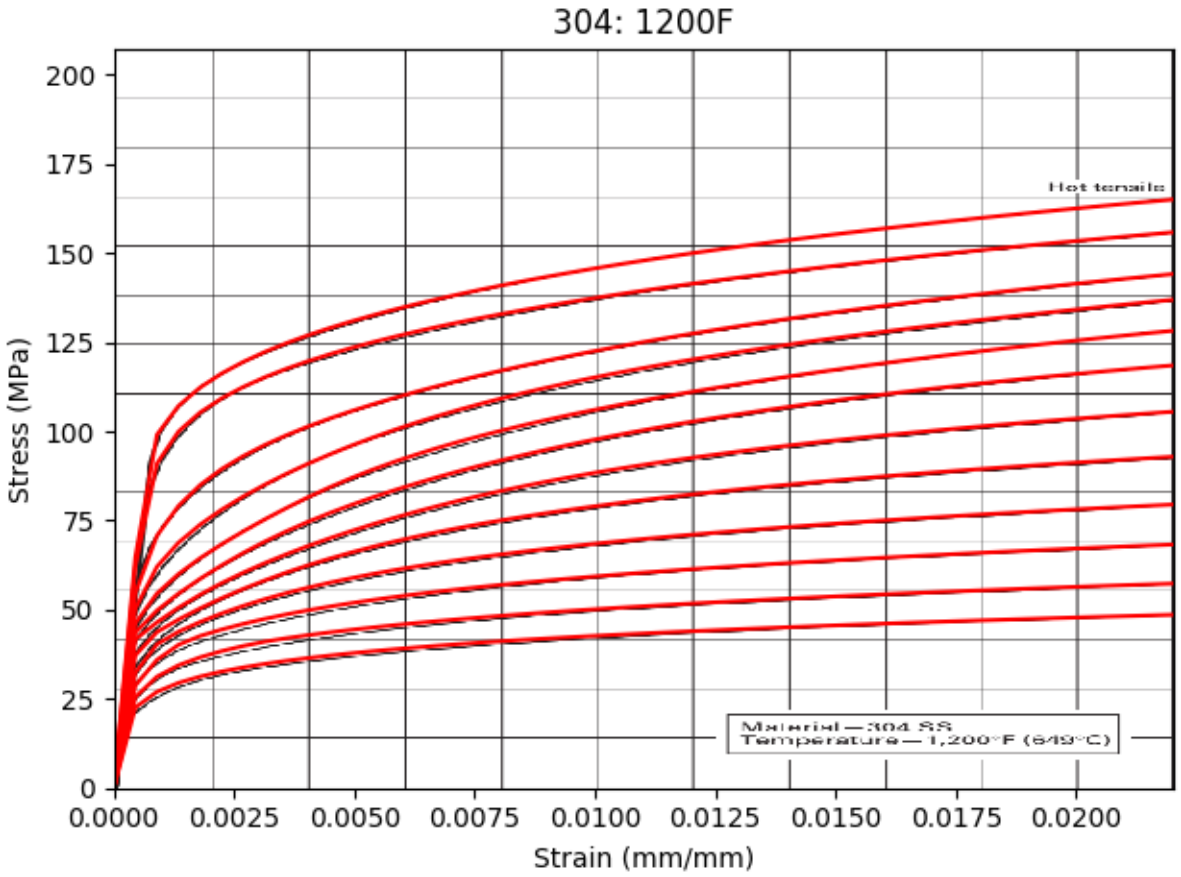

Fig. 5: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1200°F.

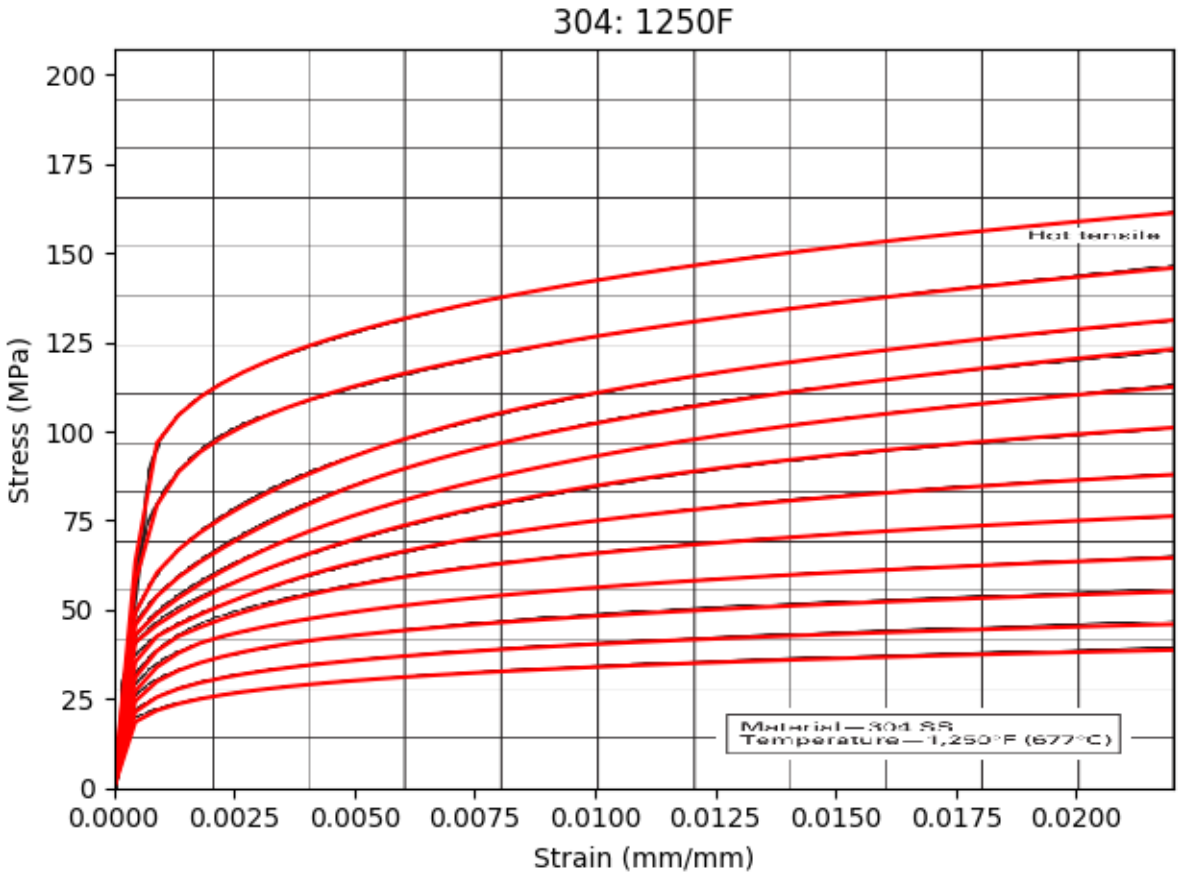

Fig. 6: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1250°F.

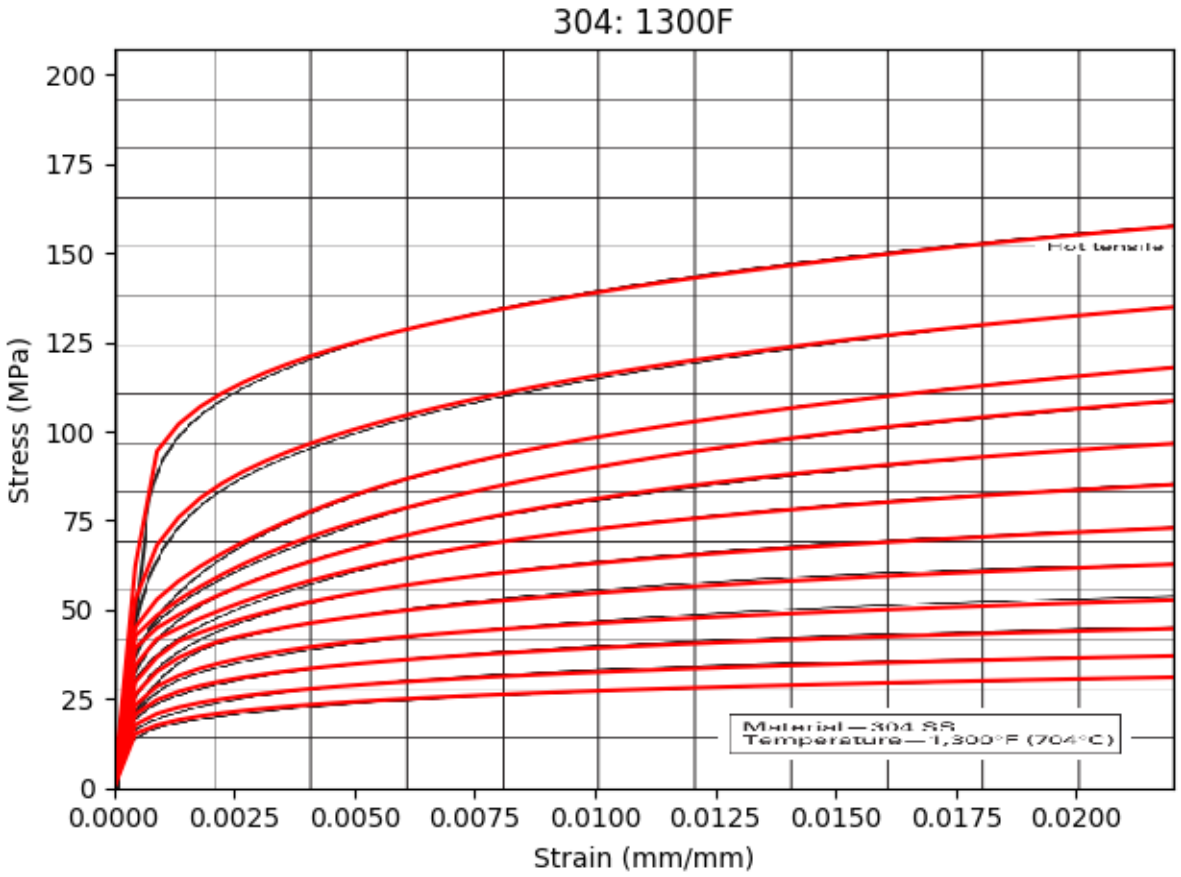

Fig. 7: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1300°F.

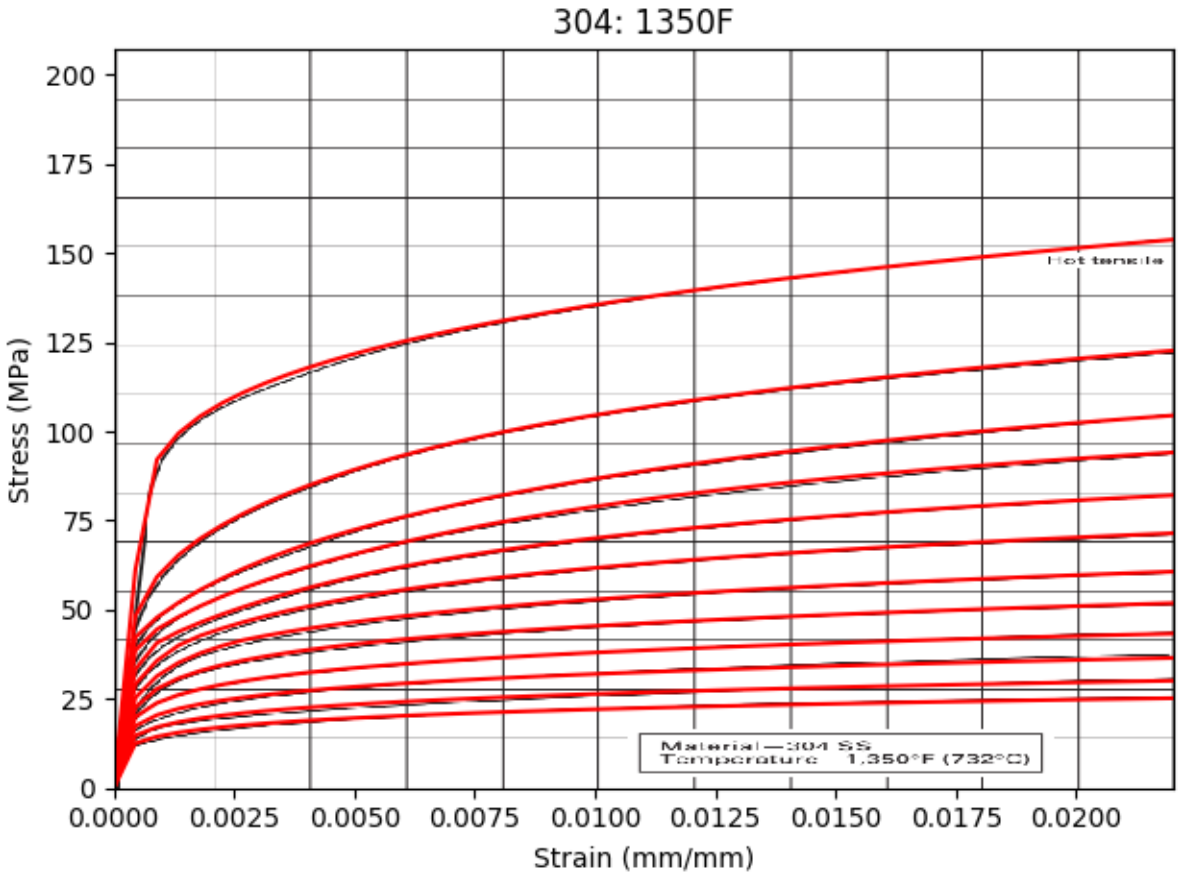

Fig. 8: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1350°F.

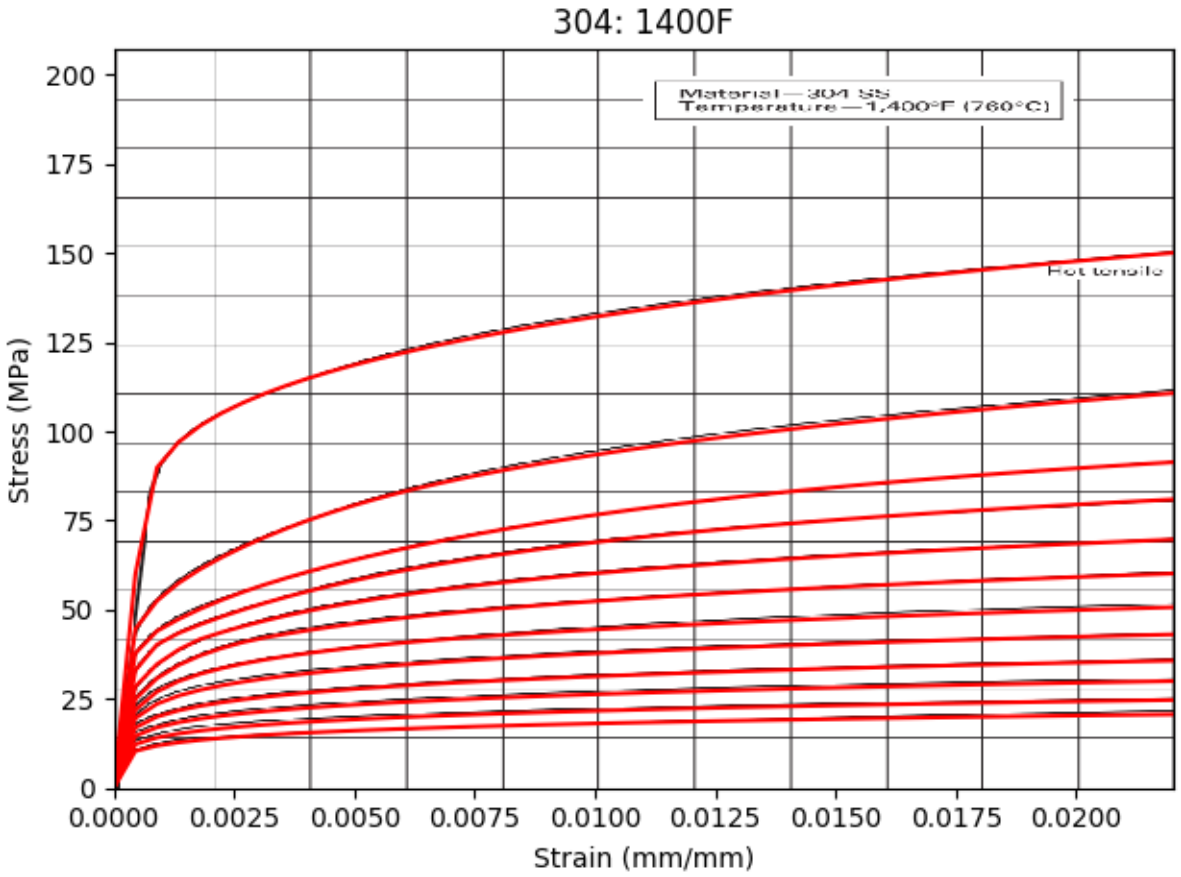

Fig. 9: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1400°F.

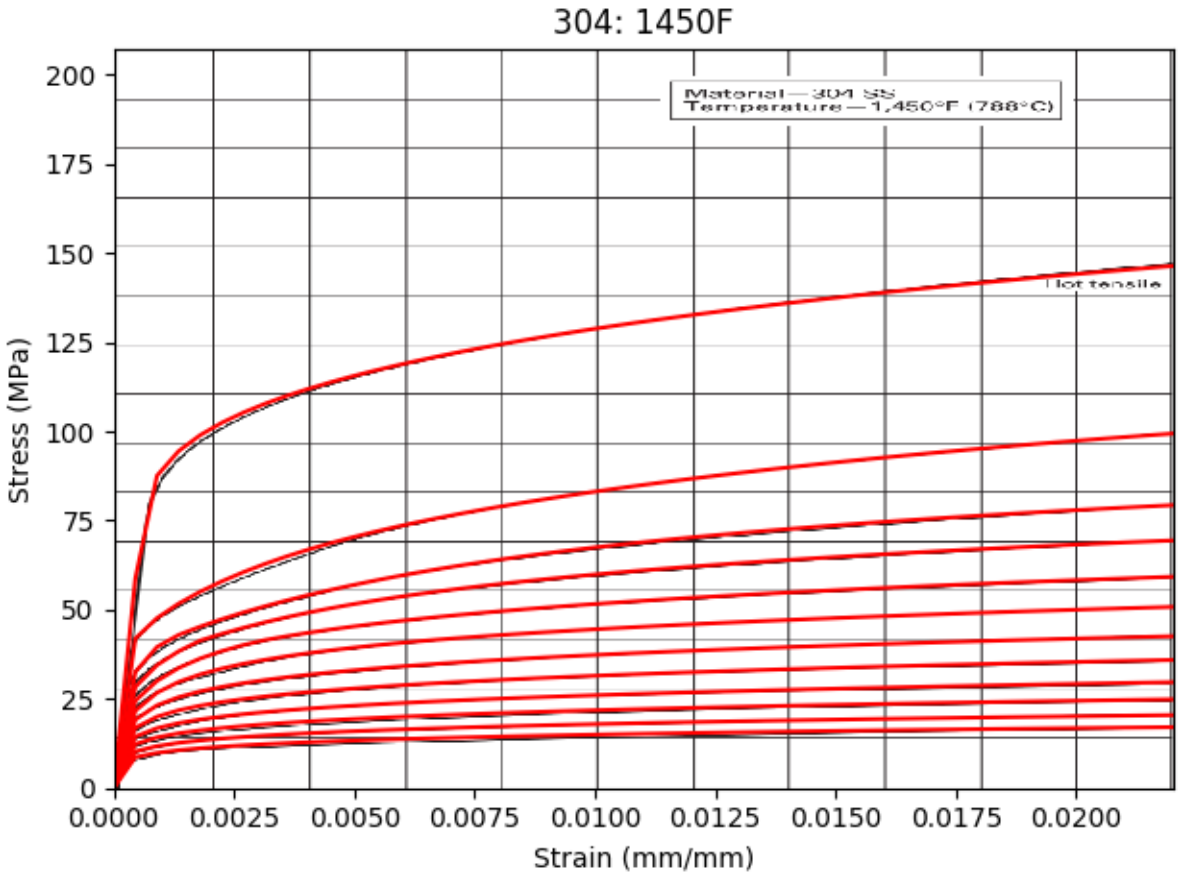

Fig. 10: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1450°F.

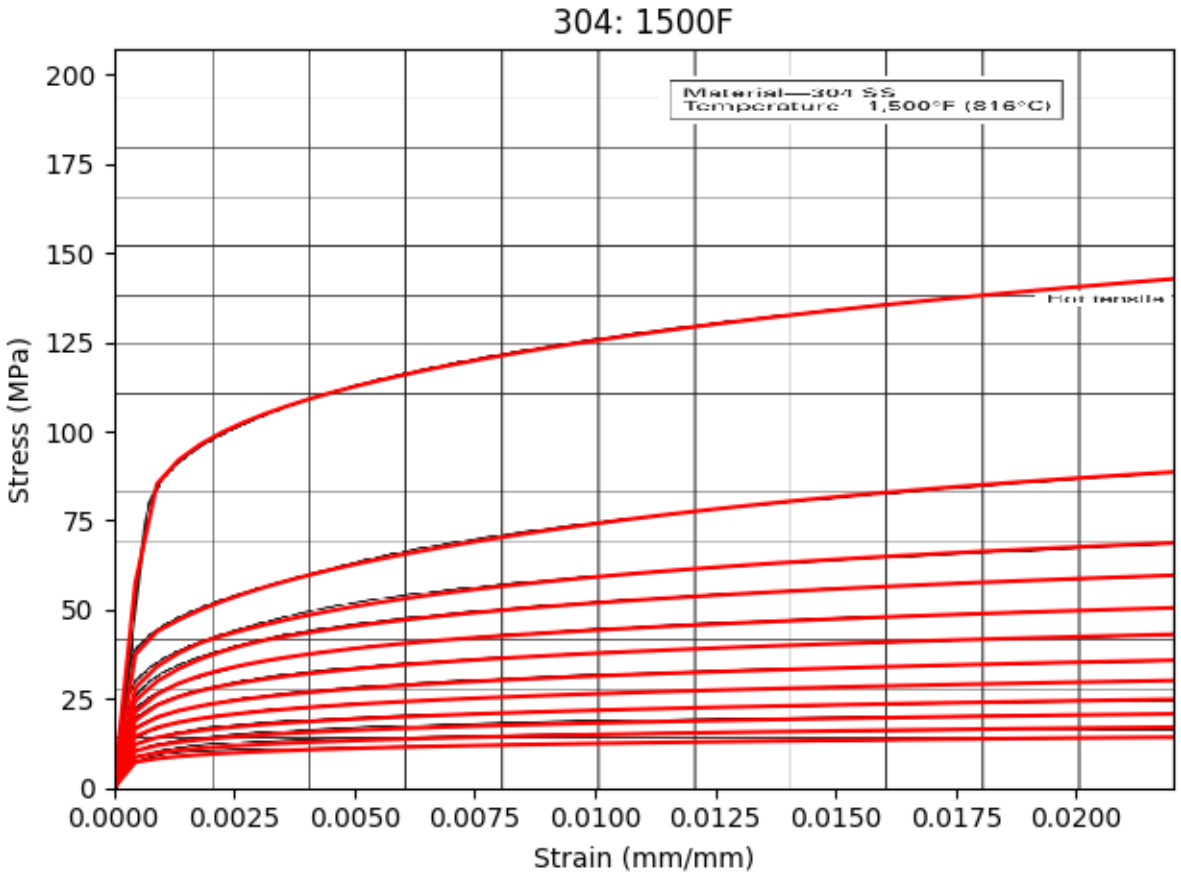

Fig. 11: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 1500°F.

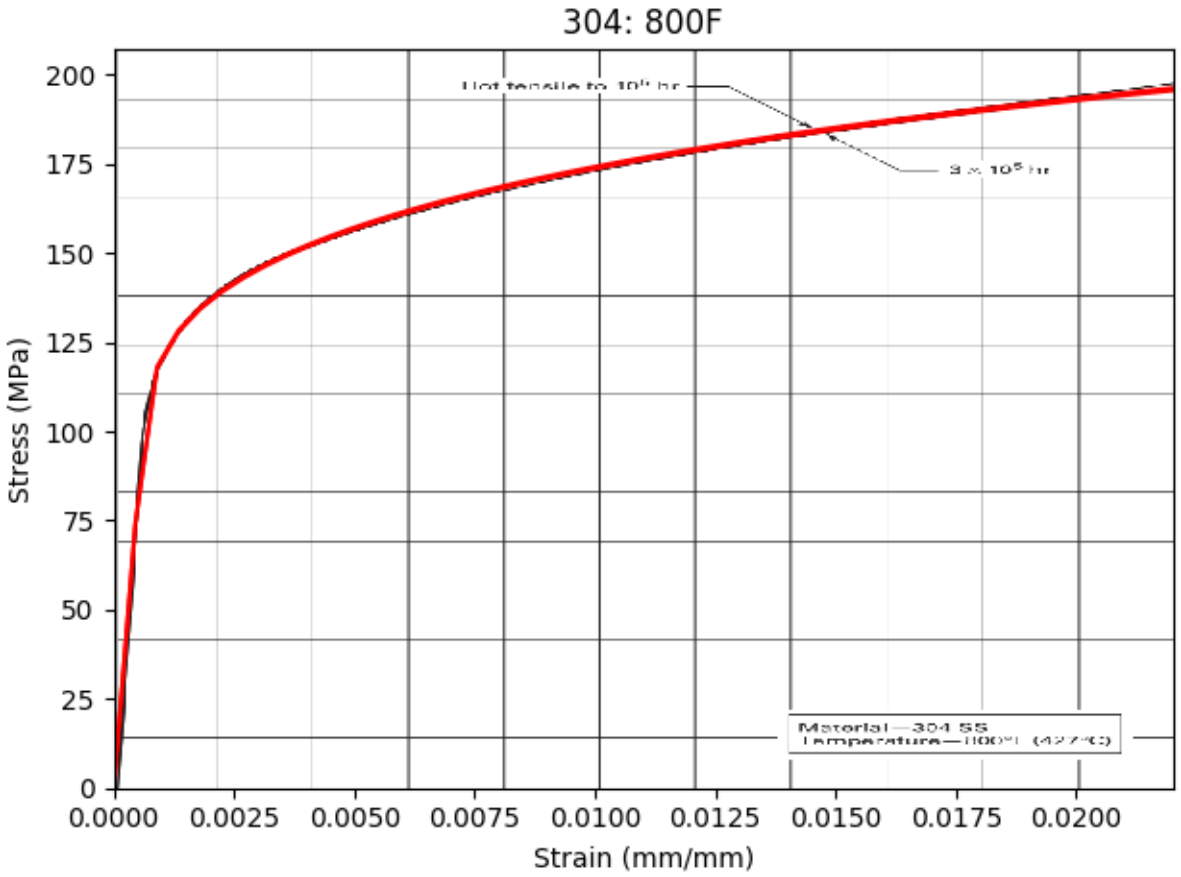

Fig. 12: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 800°F.

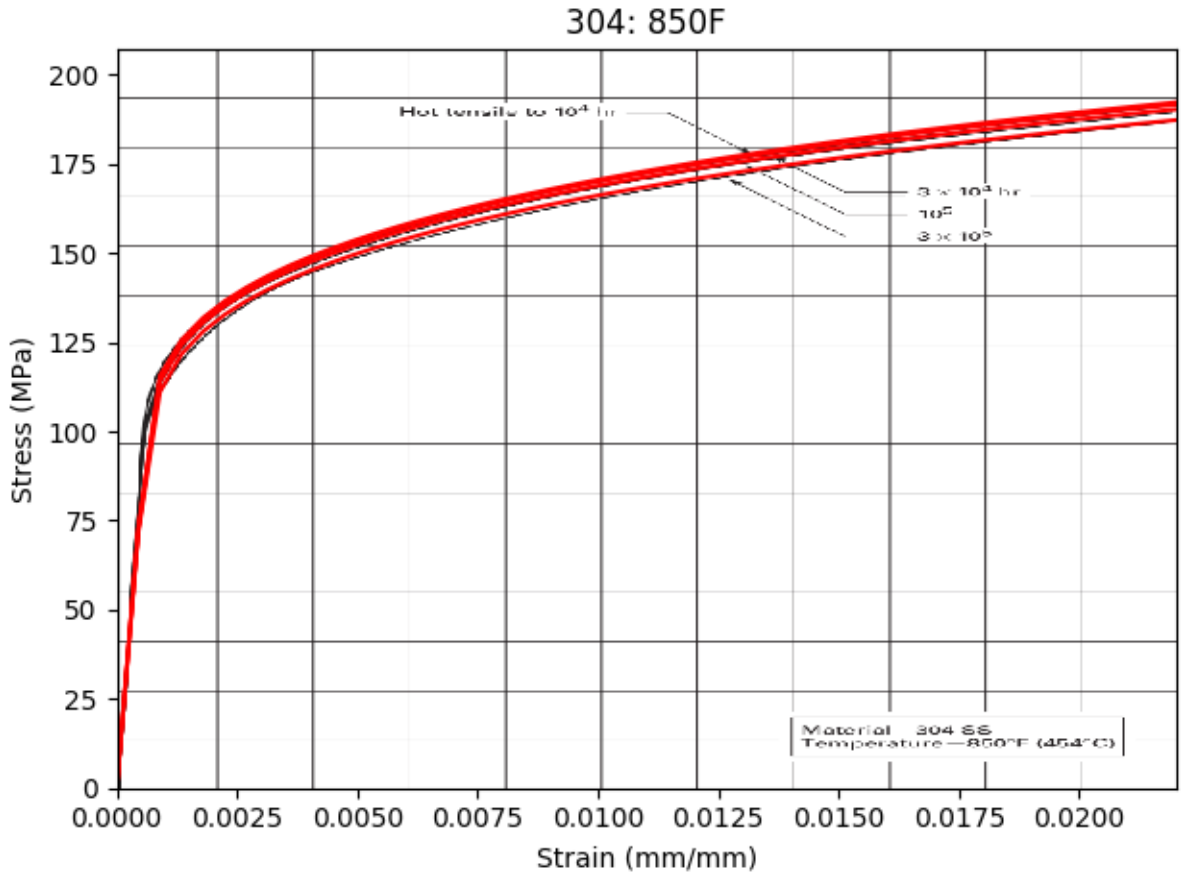

Fig. 13: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 850°F.

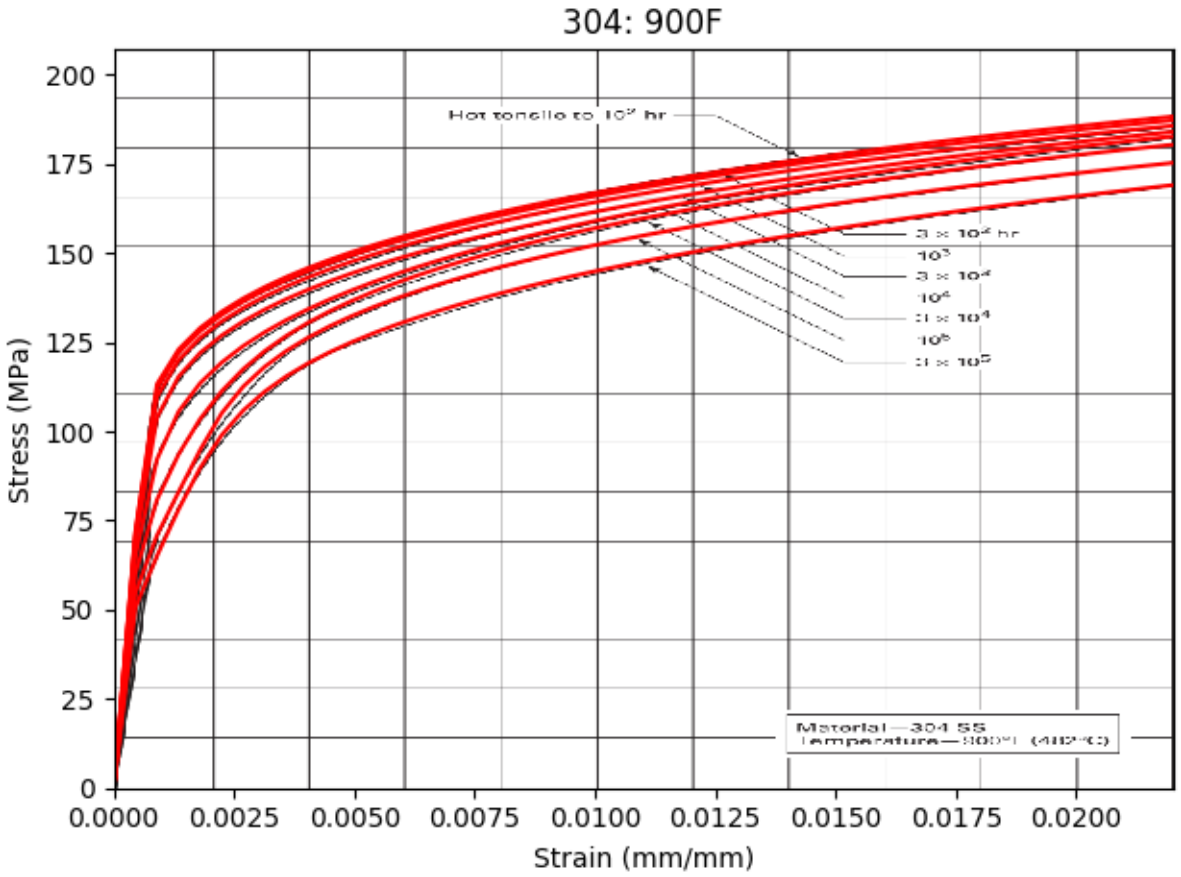

Fig. 14: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 900°F.

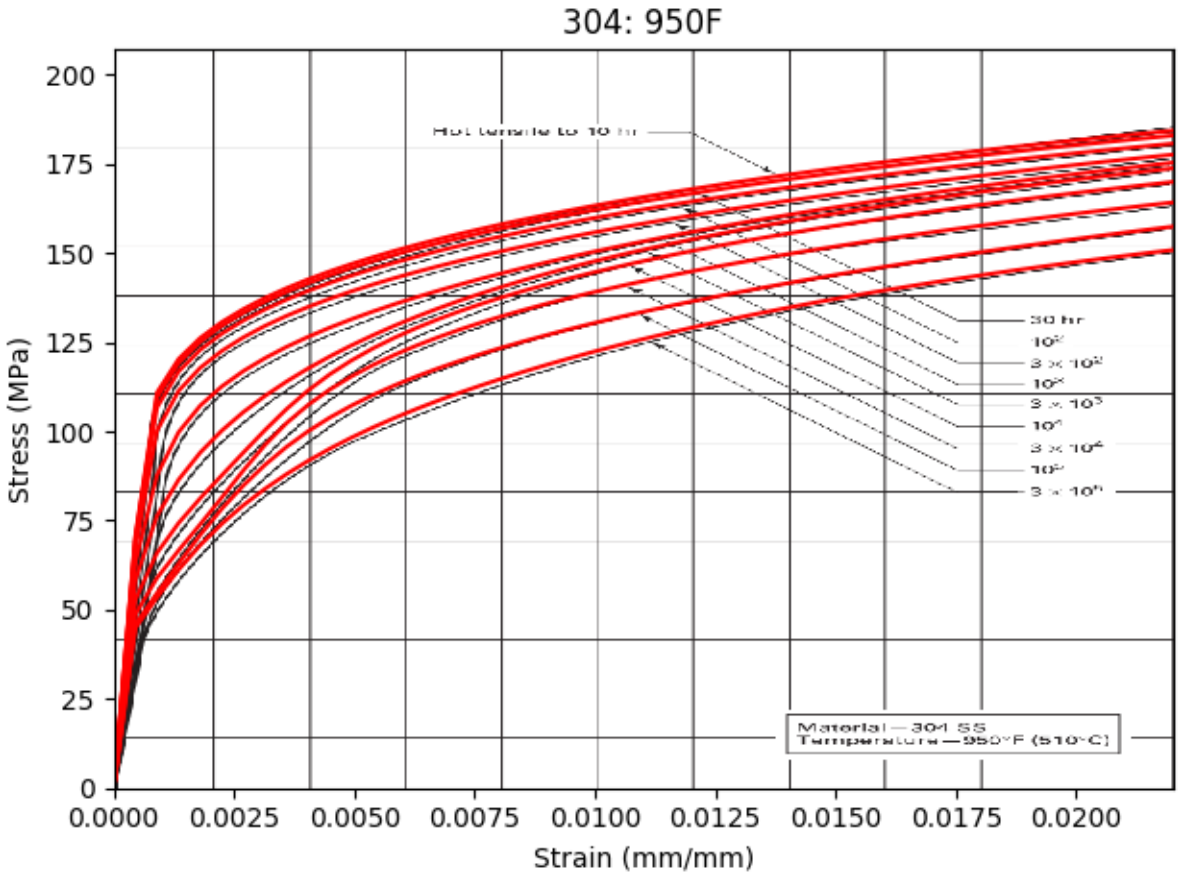

Fig. 15: Comparison between the current and implemented isochronous stress-strain curves for 304SS at 950°F.

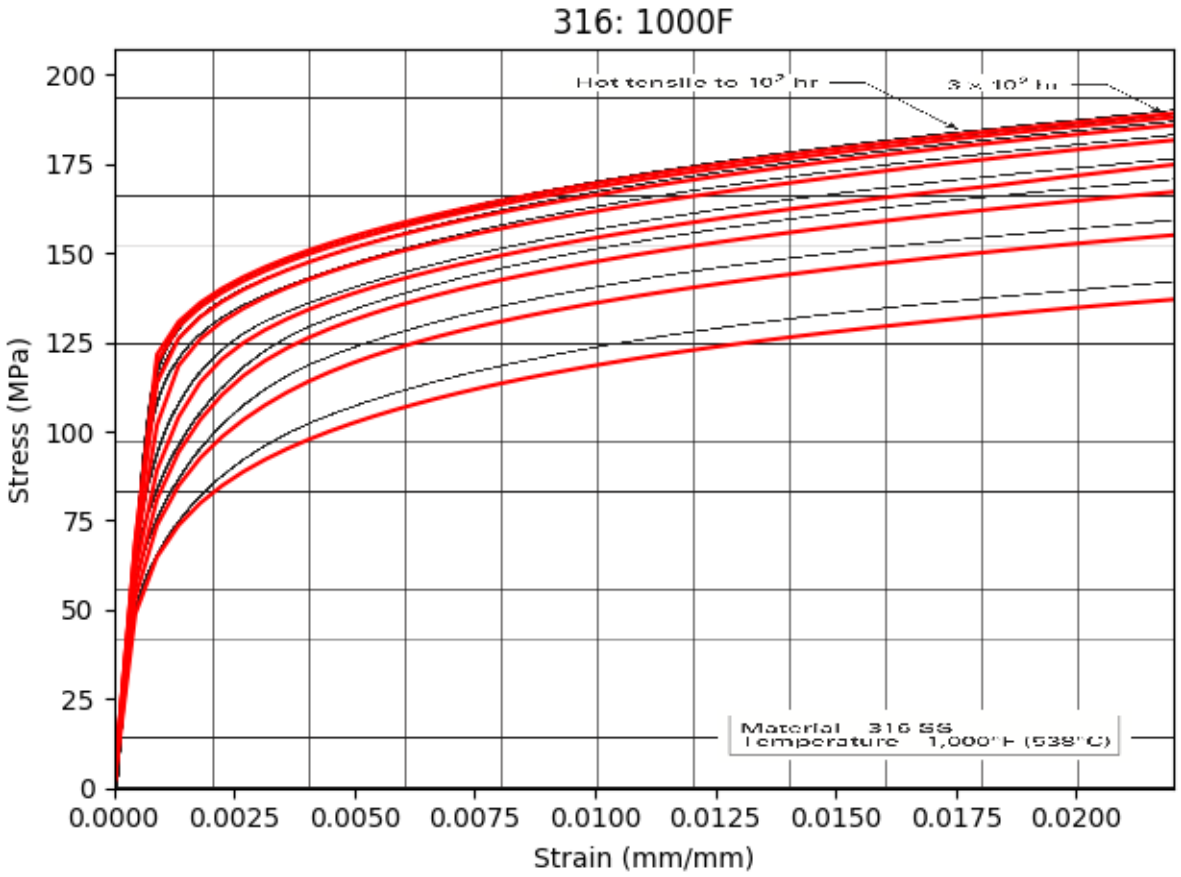

Fig. 16: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1000°F.

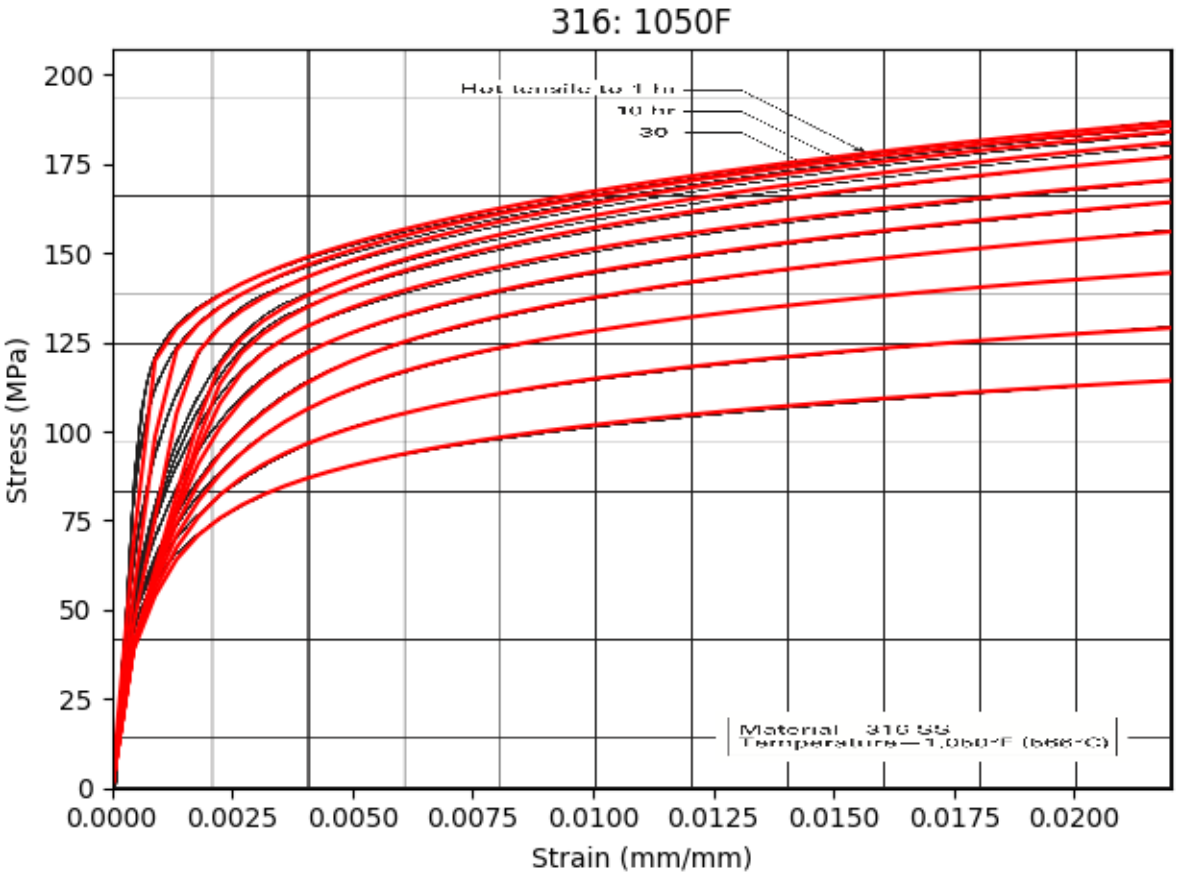

Fig. 17: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1050°F.

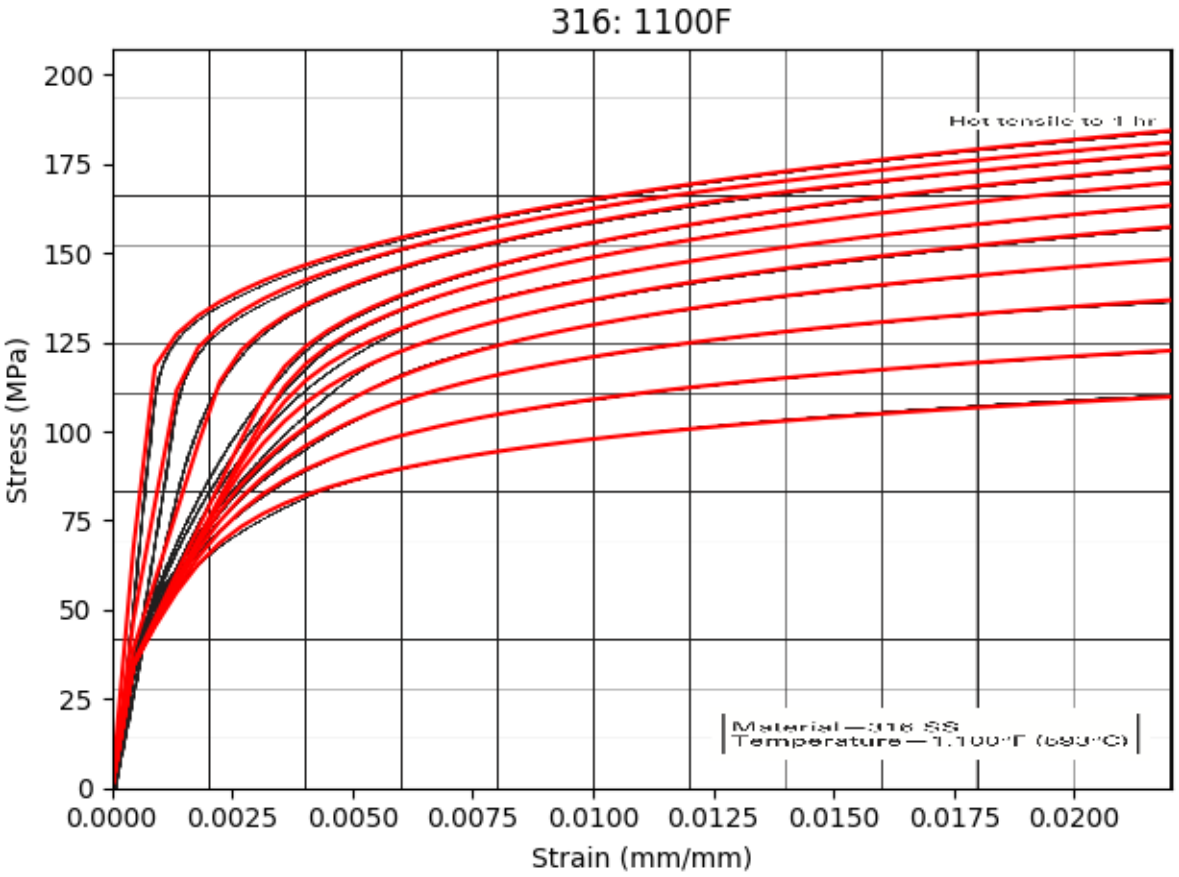

Fig. 18: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1100°F.

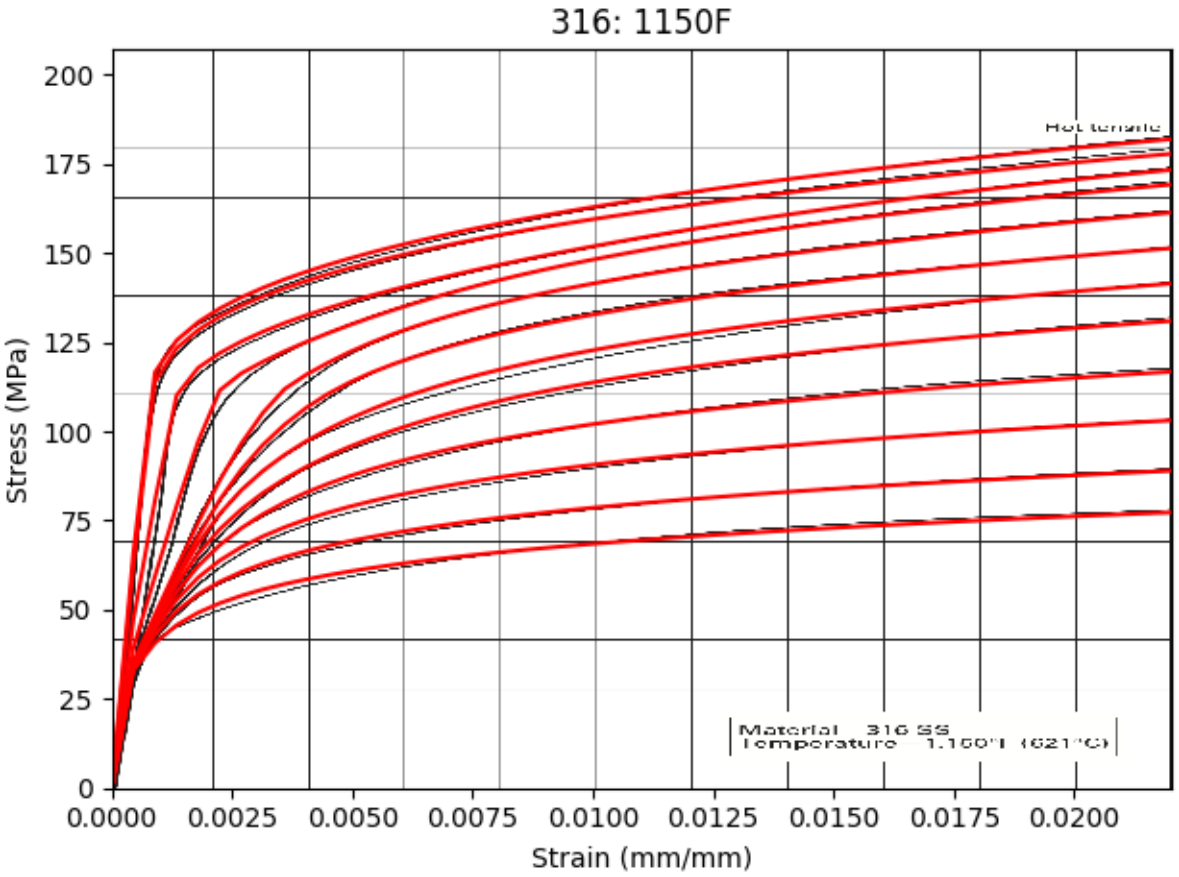

Fig. 19: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1150°F.

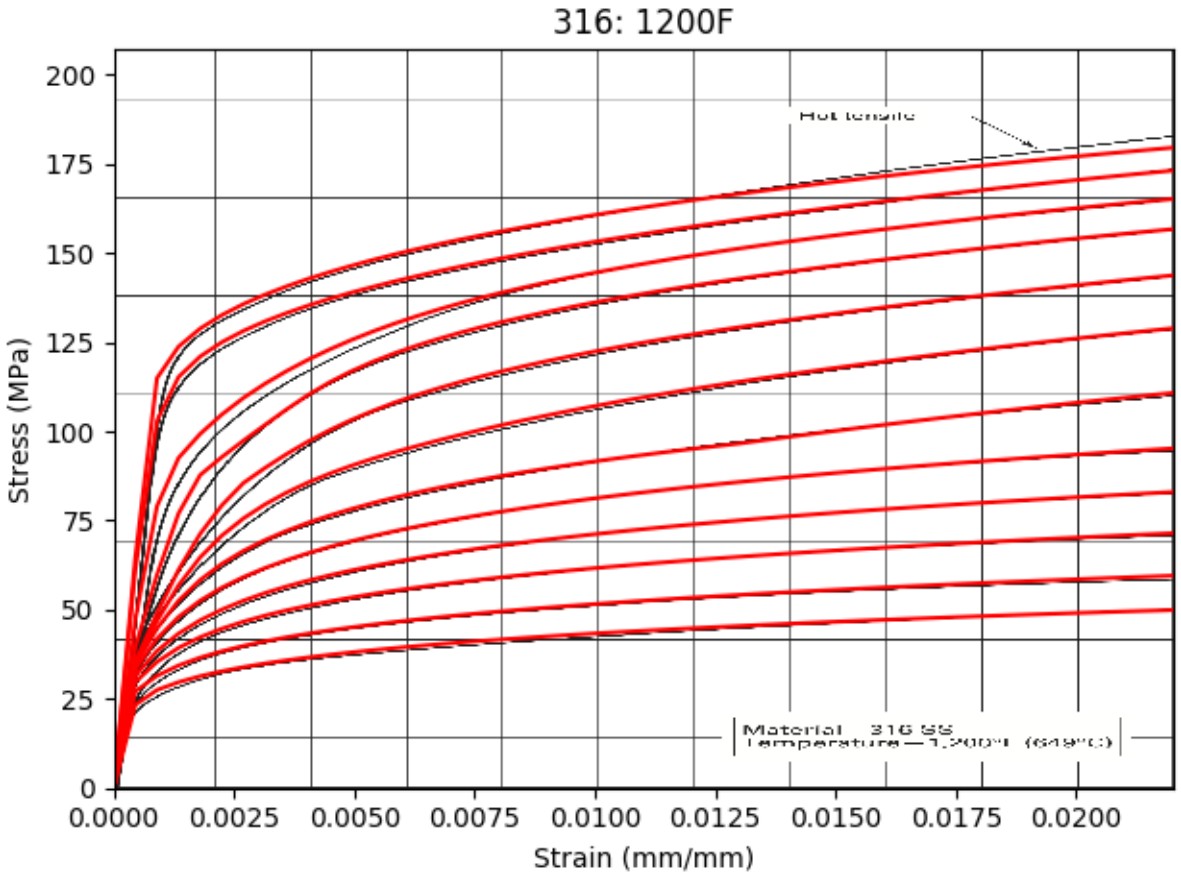

Fig. 20: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1200°F.

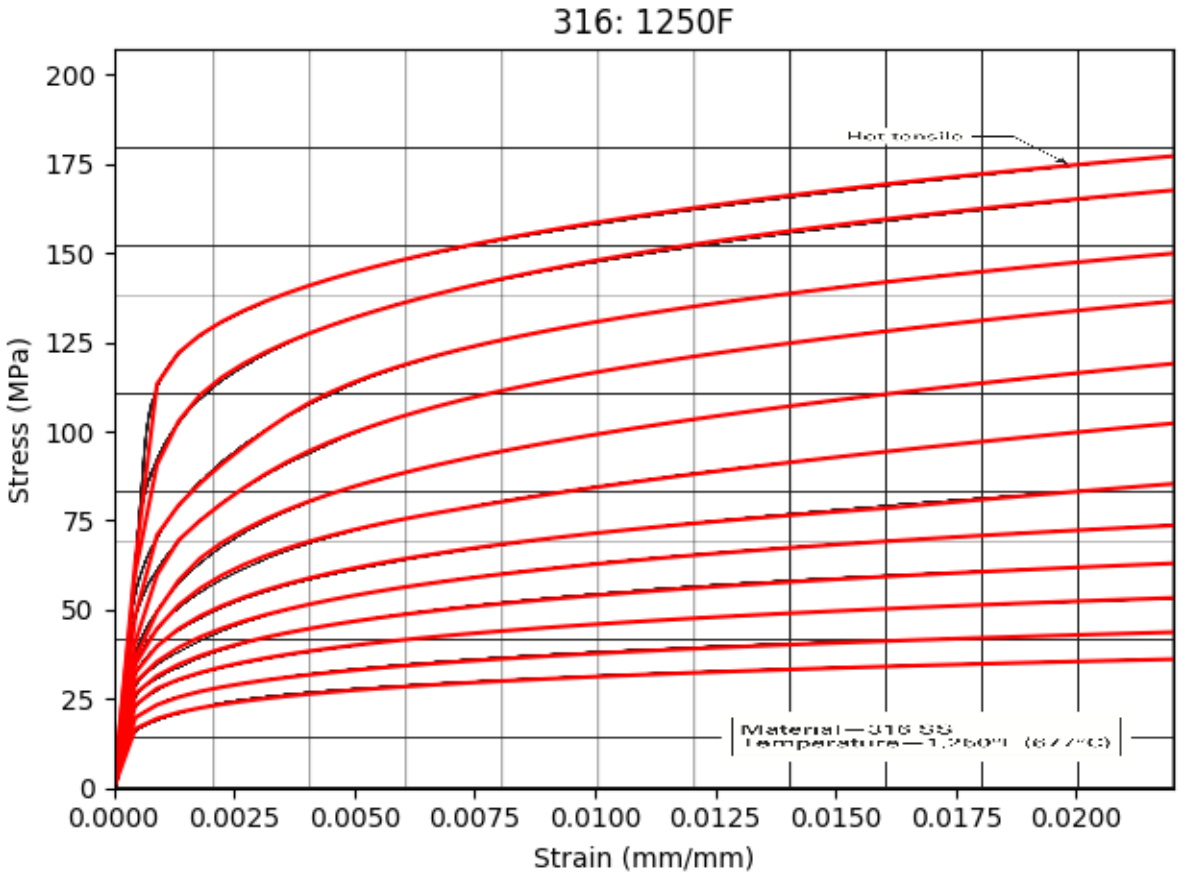

Fig. 21: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1250°F.

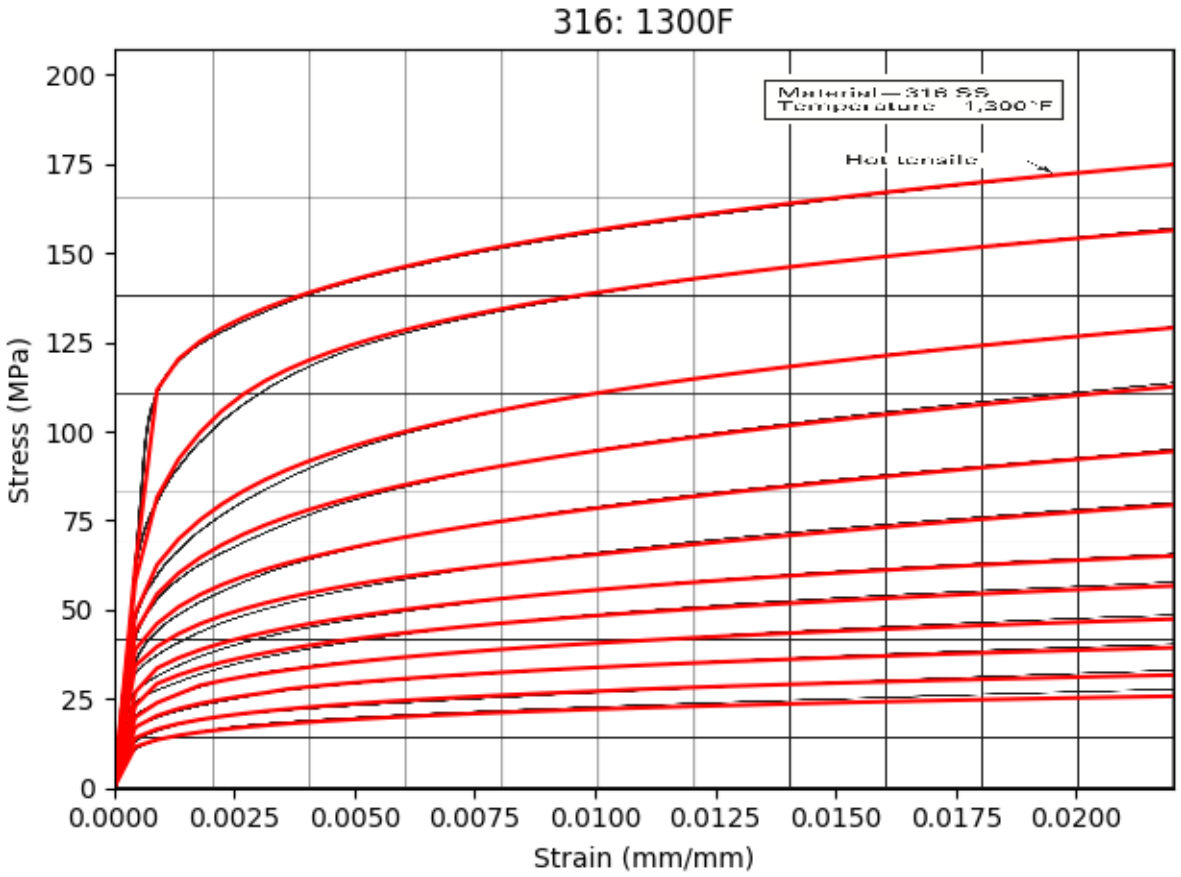

Fig. 22: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1300°F.

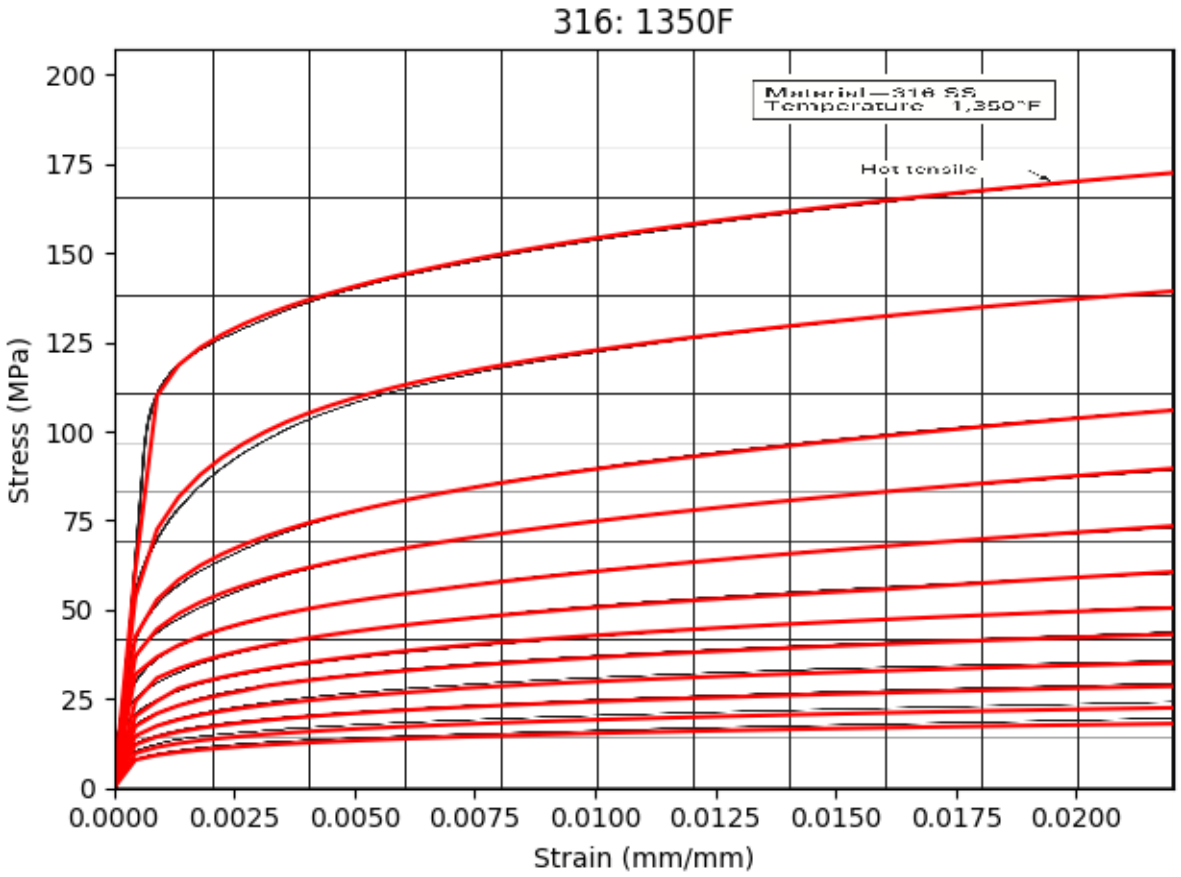

Fig. 23: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1350°F.

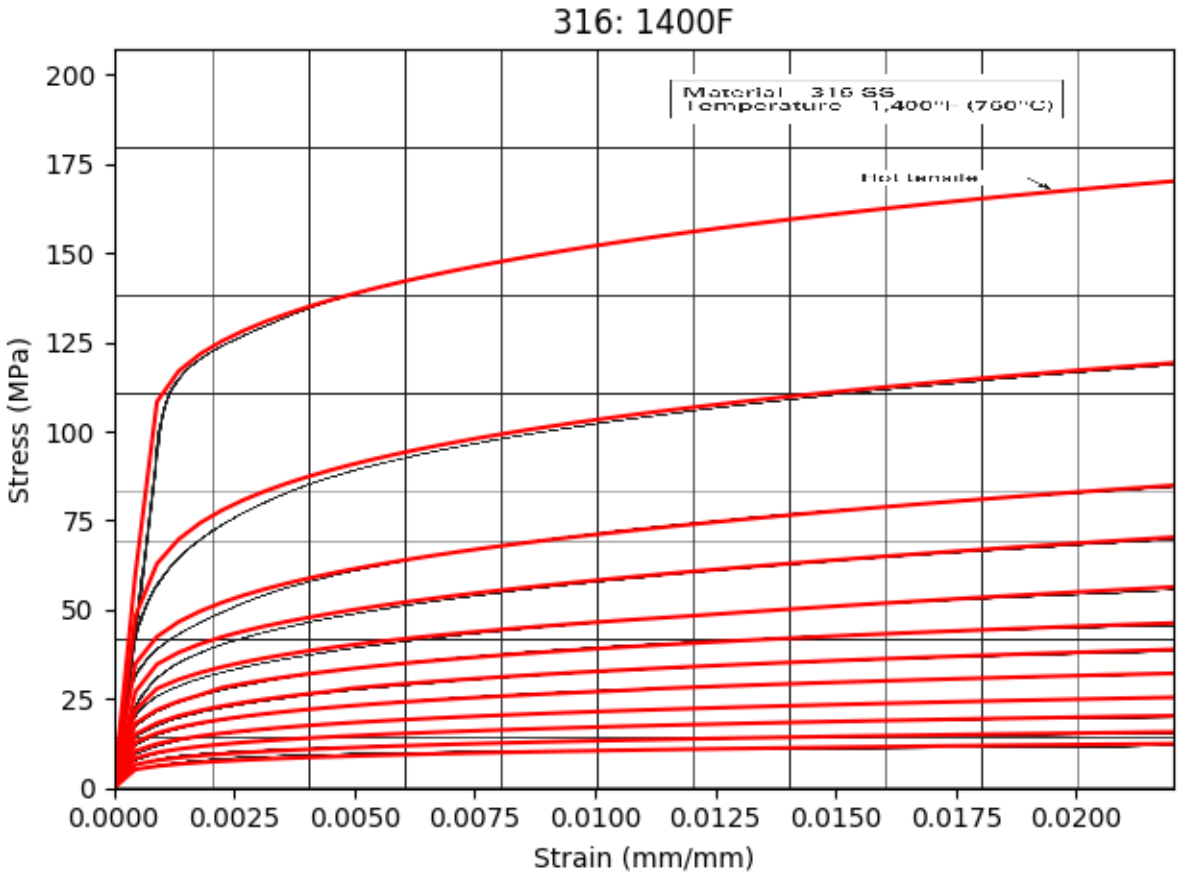

Fig. 24: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1400°F.

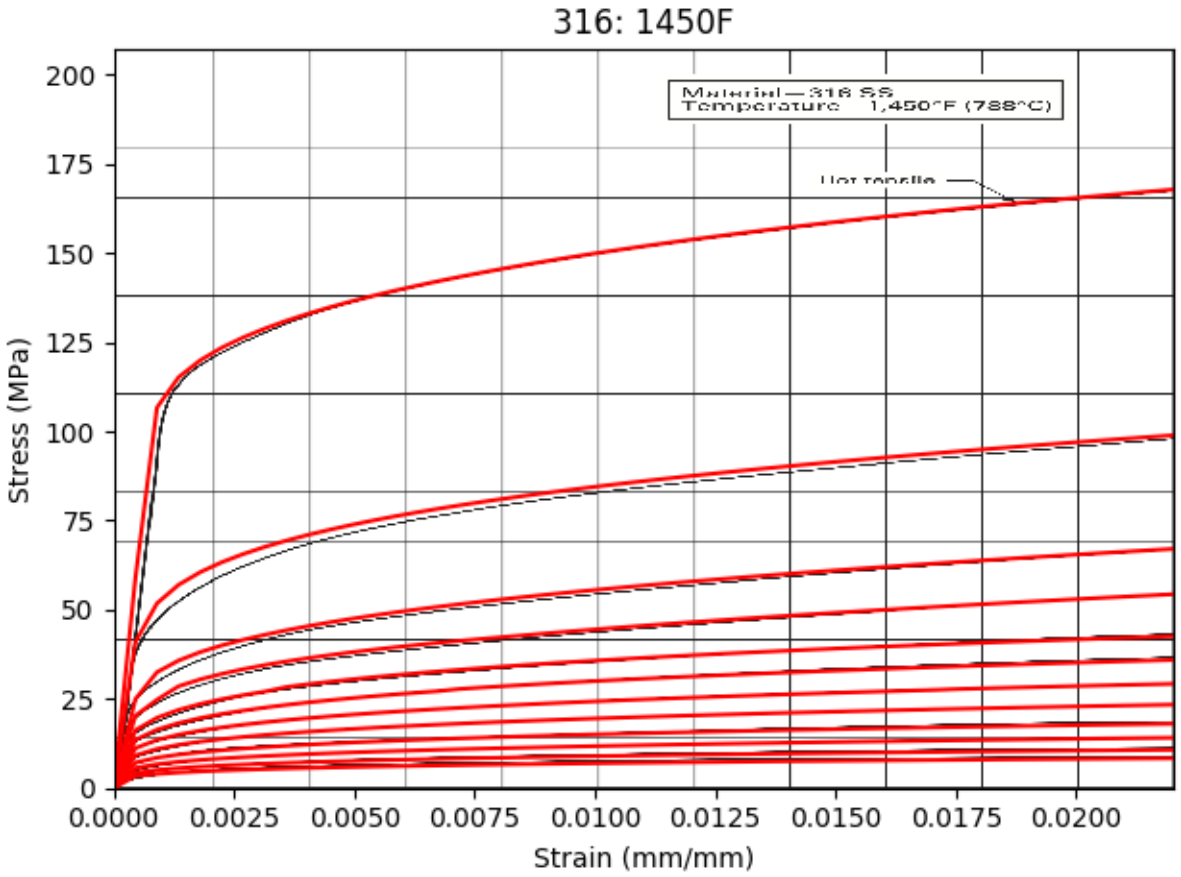

Fig. 25: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1450°F.

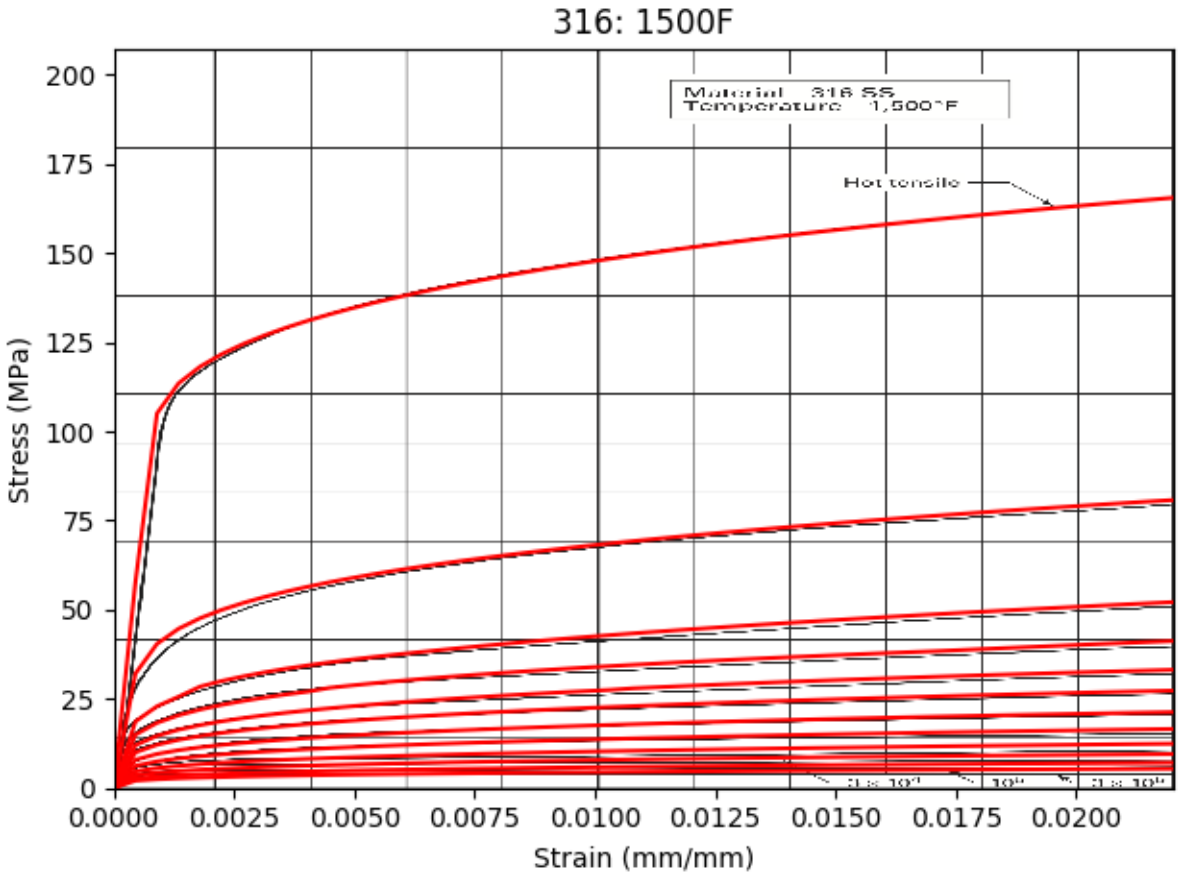

Fig. 26: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 1500°F.

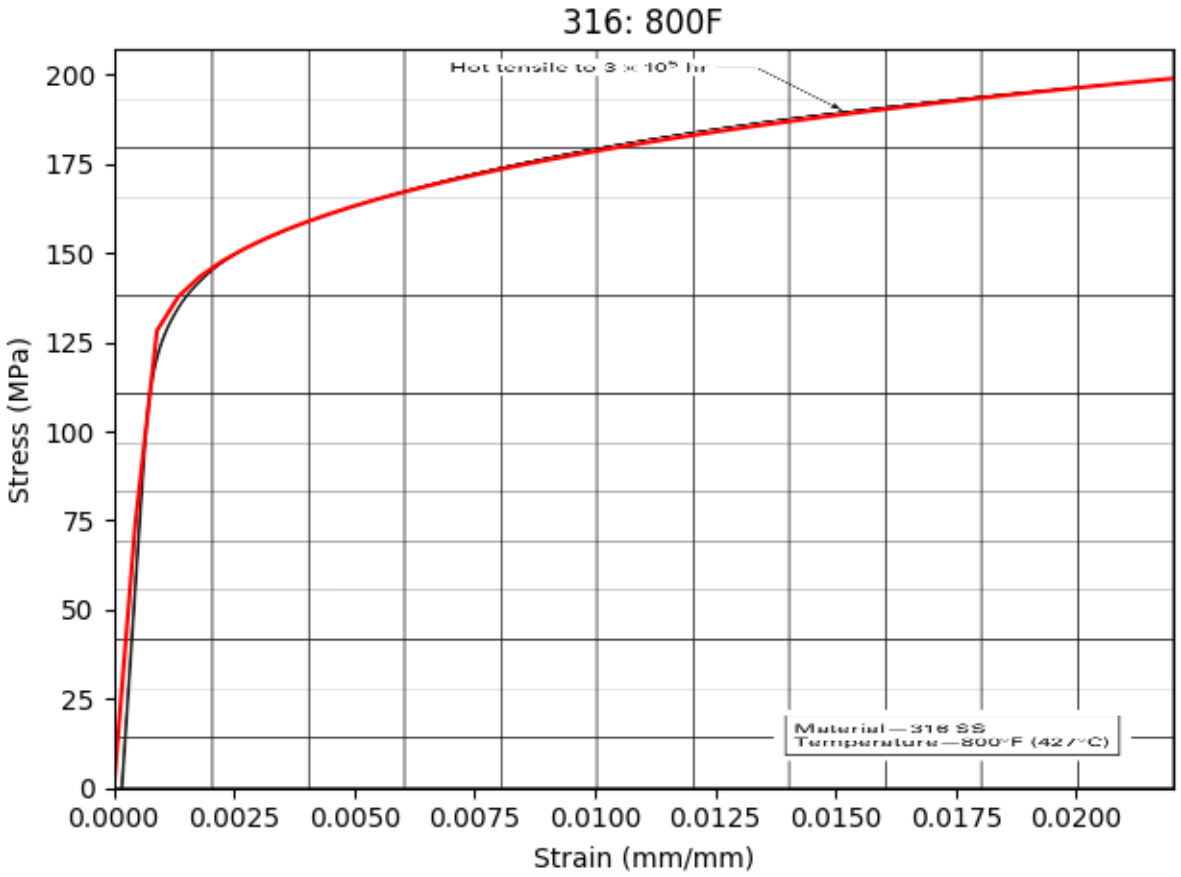

Fig. 27: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 800°F.

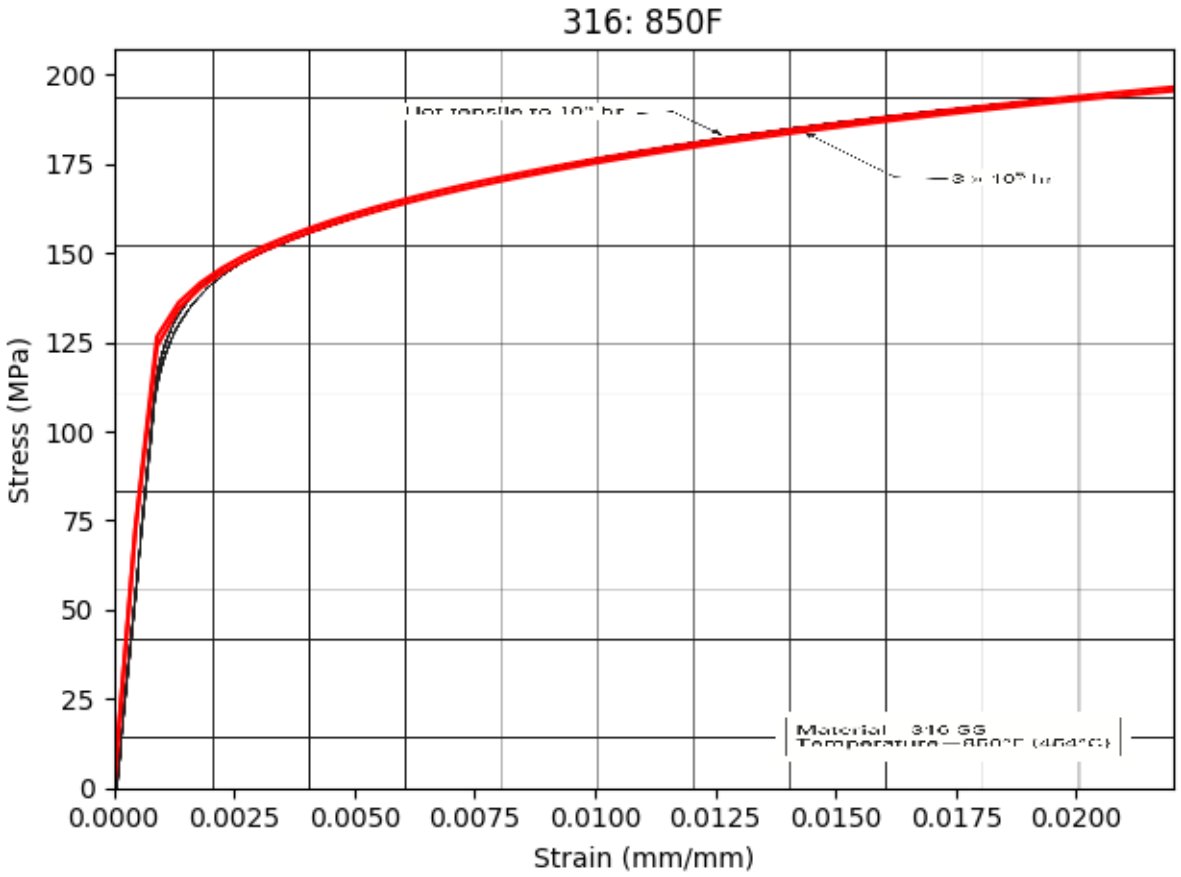

Fig. 28: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 850°F.

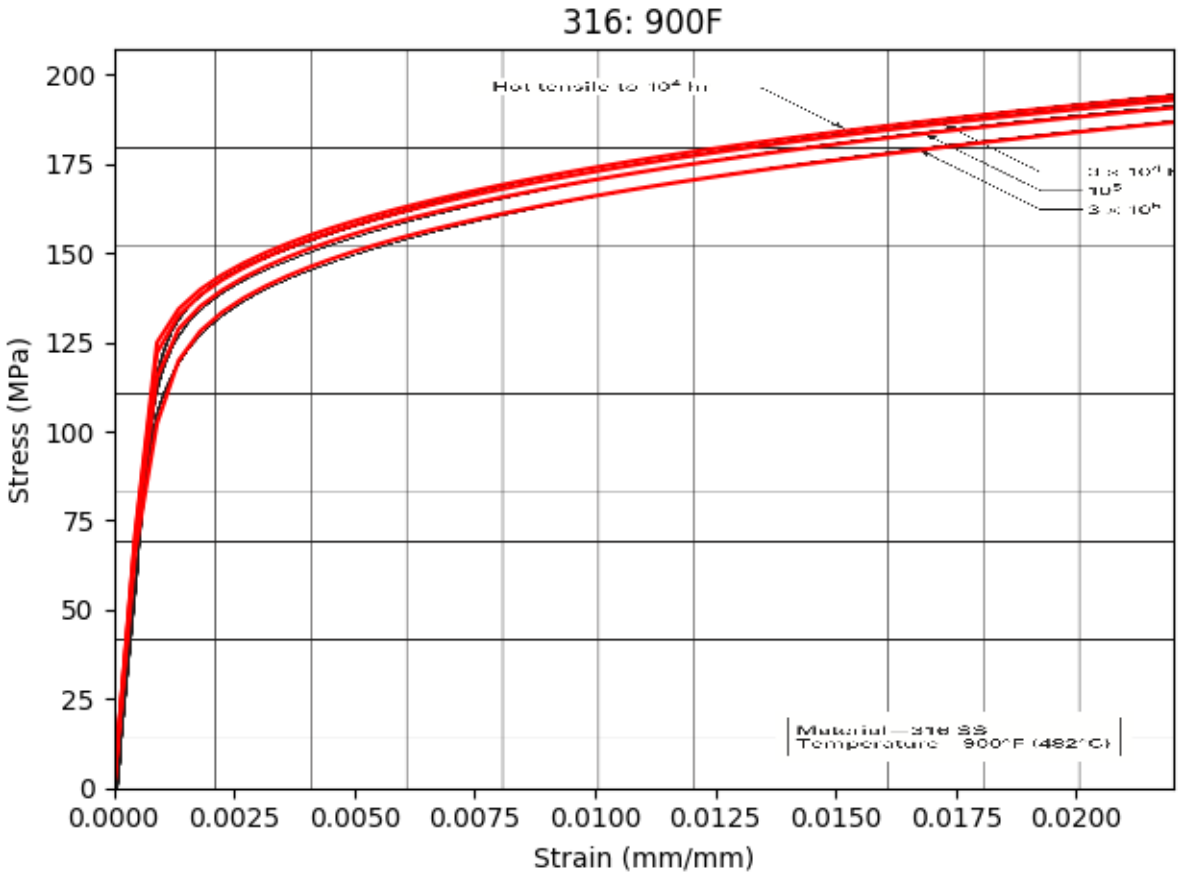

Fig. 29: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 900°F.

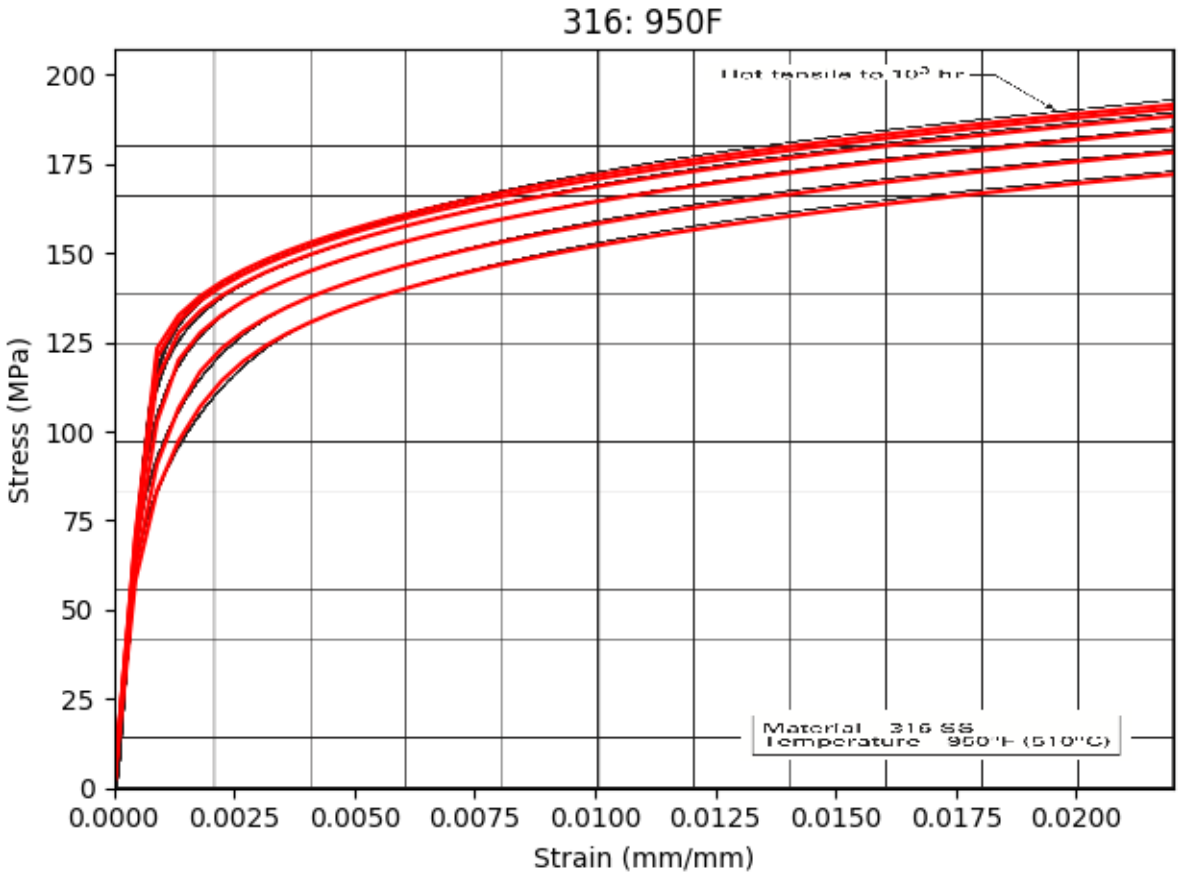

Fig. 30: Comparison between the current and implemented isochronous stress-strain curves for 316SS at 950°F.

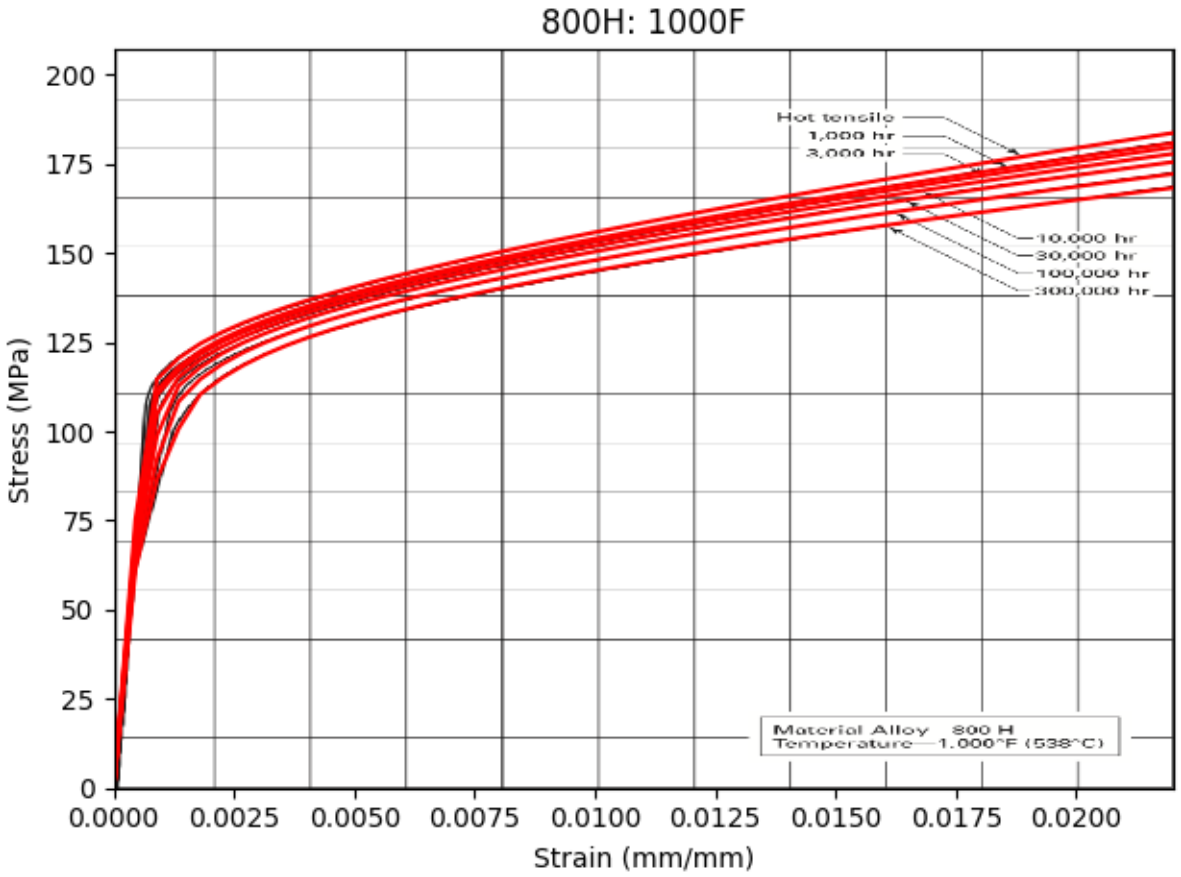

Fig. 31: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1000°F.

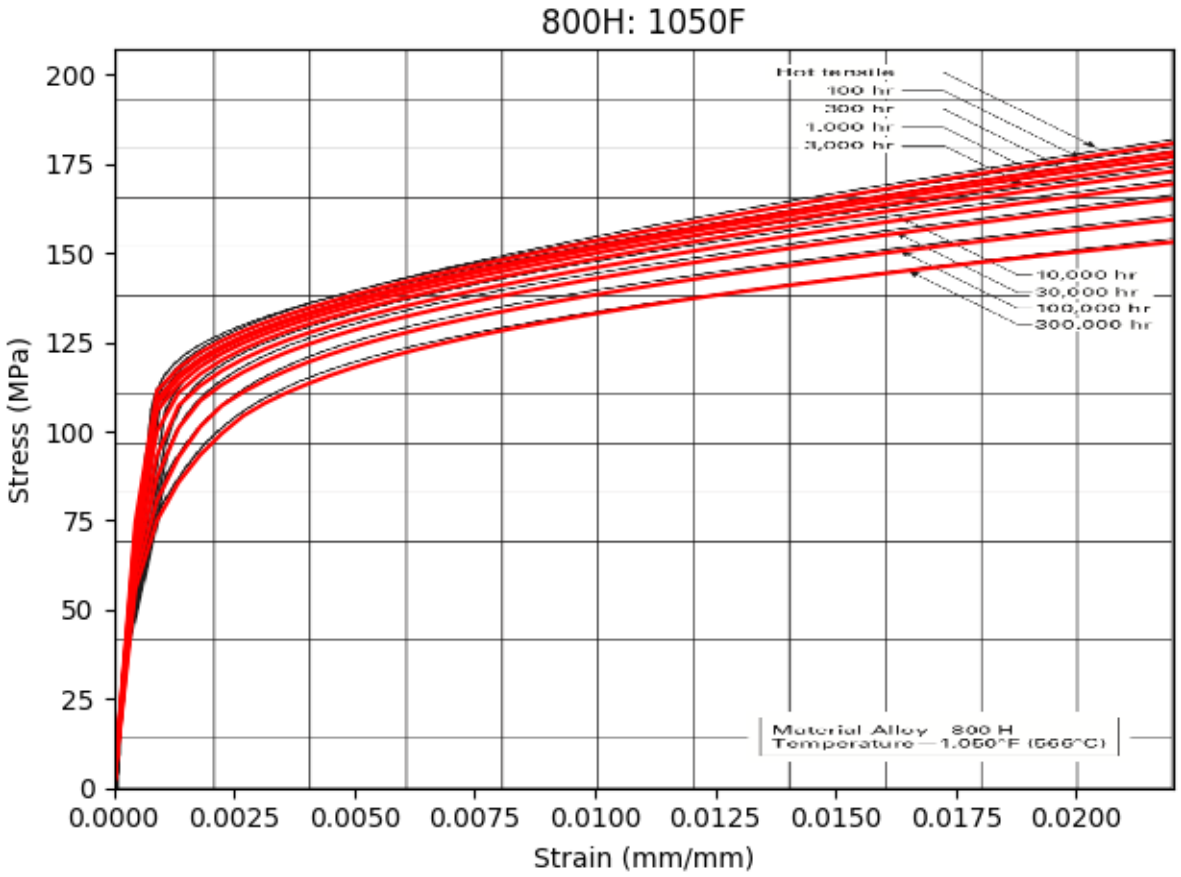

Fig. 32: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1050°F.

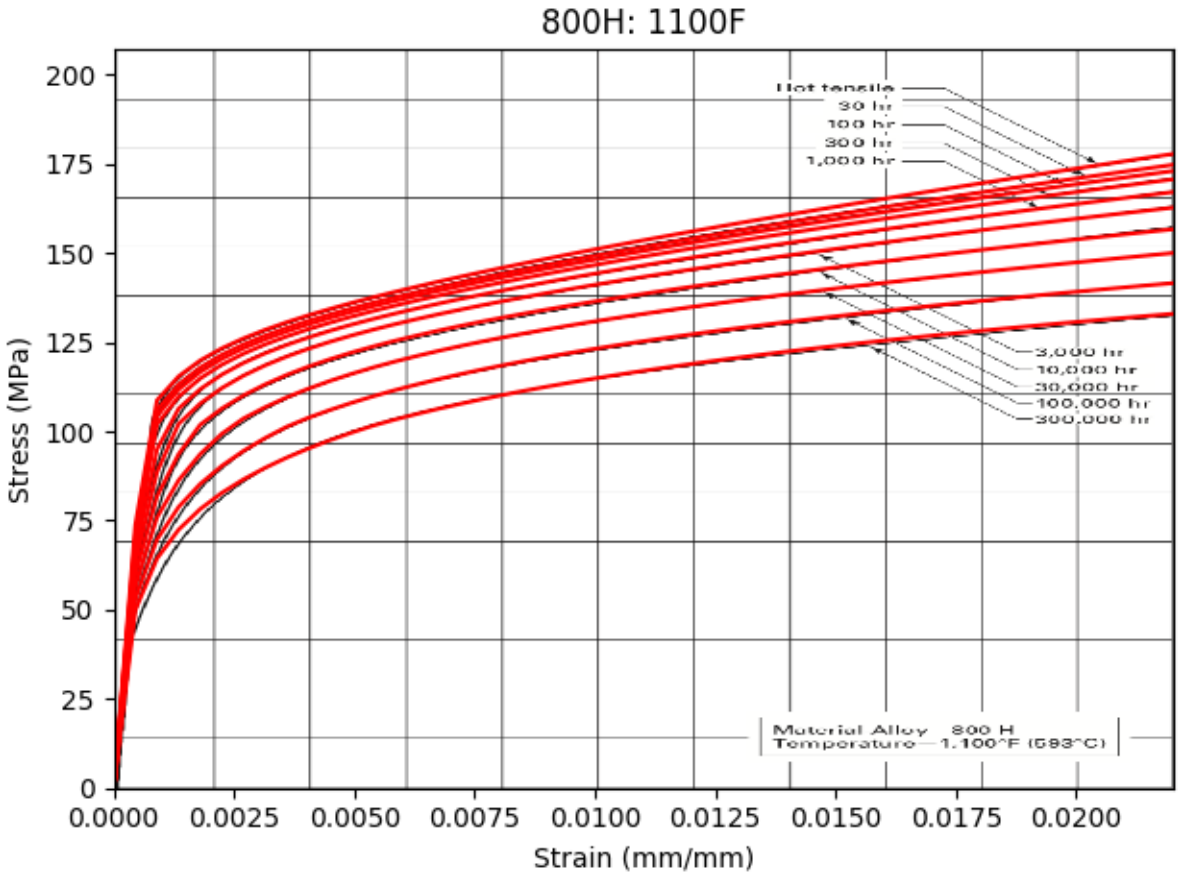

Fig. 33: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1100°F.

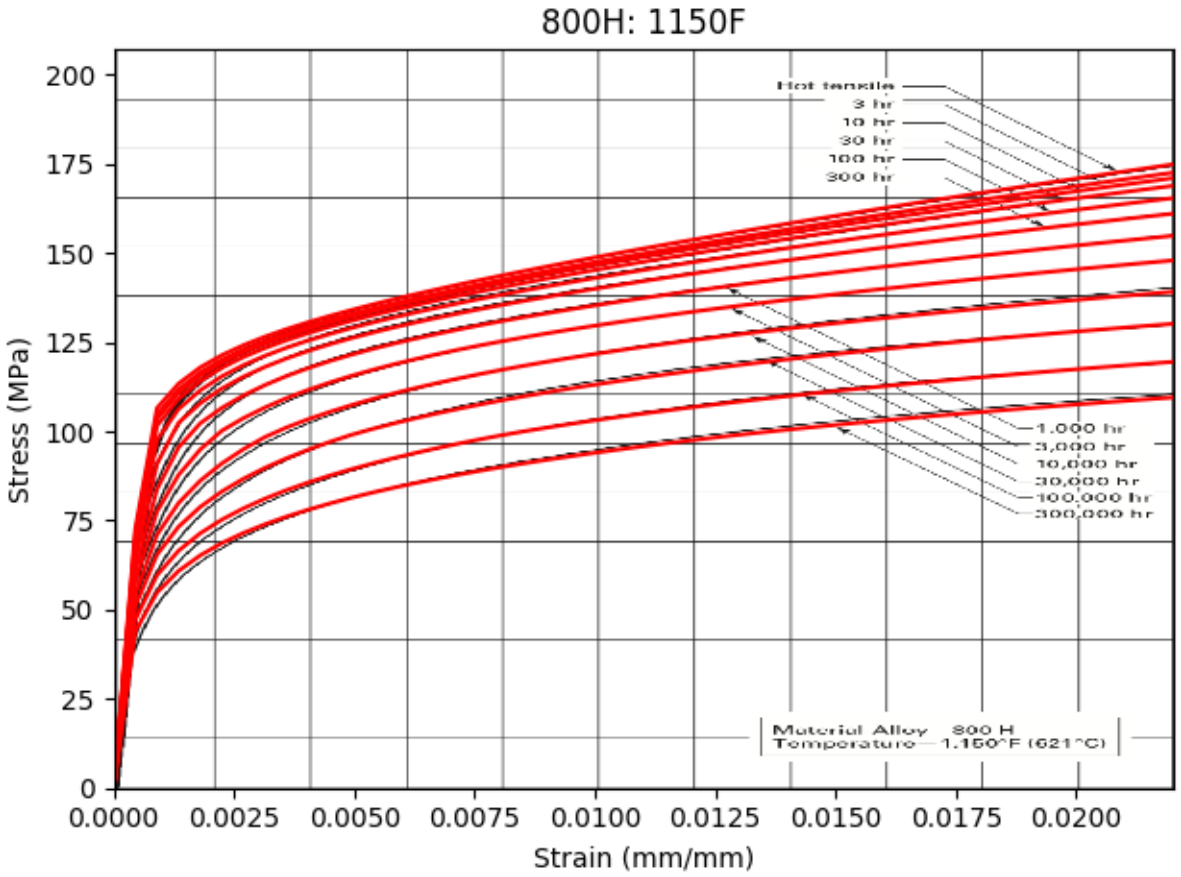

Fig. 34: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1150°F.

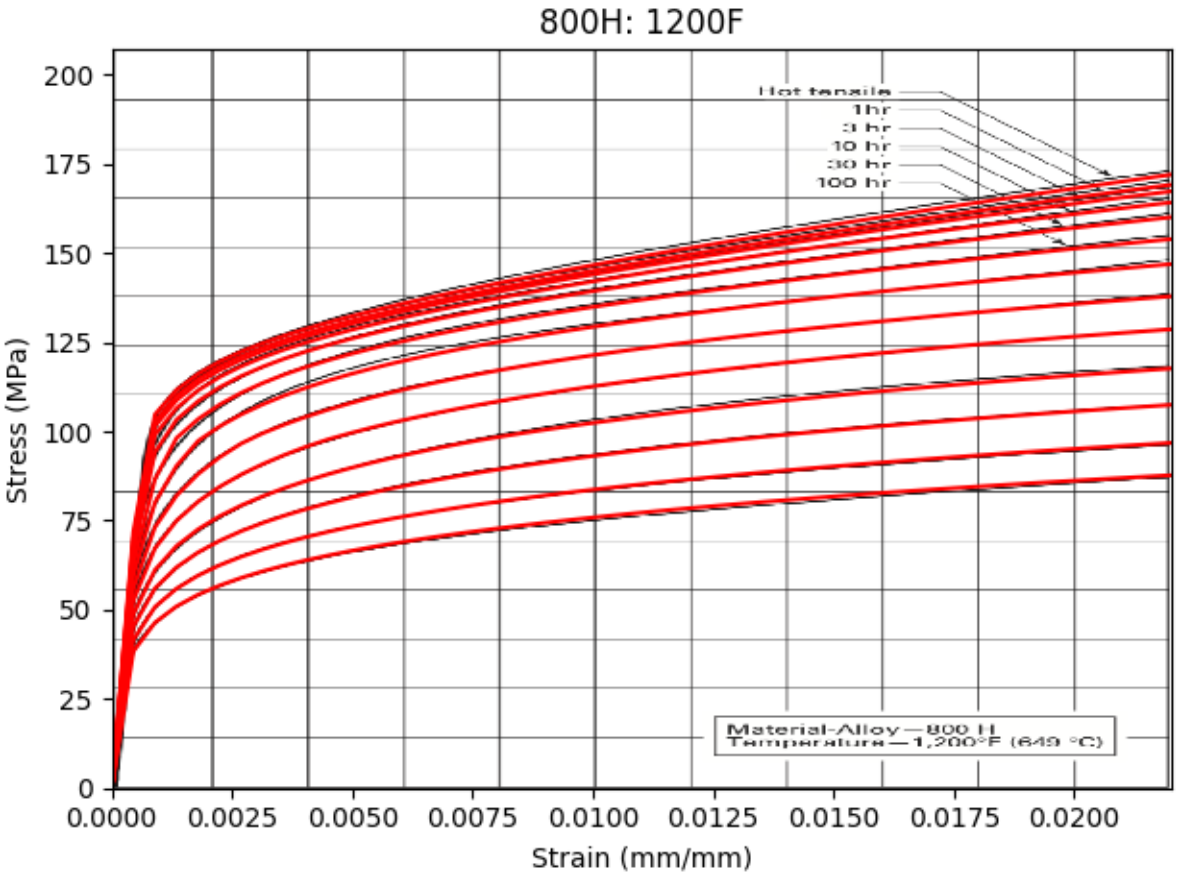

Fig. 35: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1200°F.

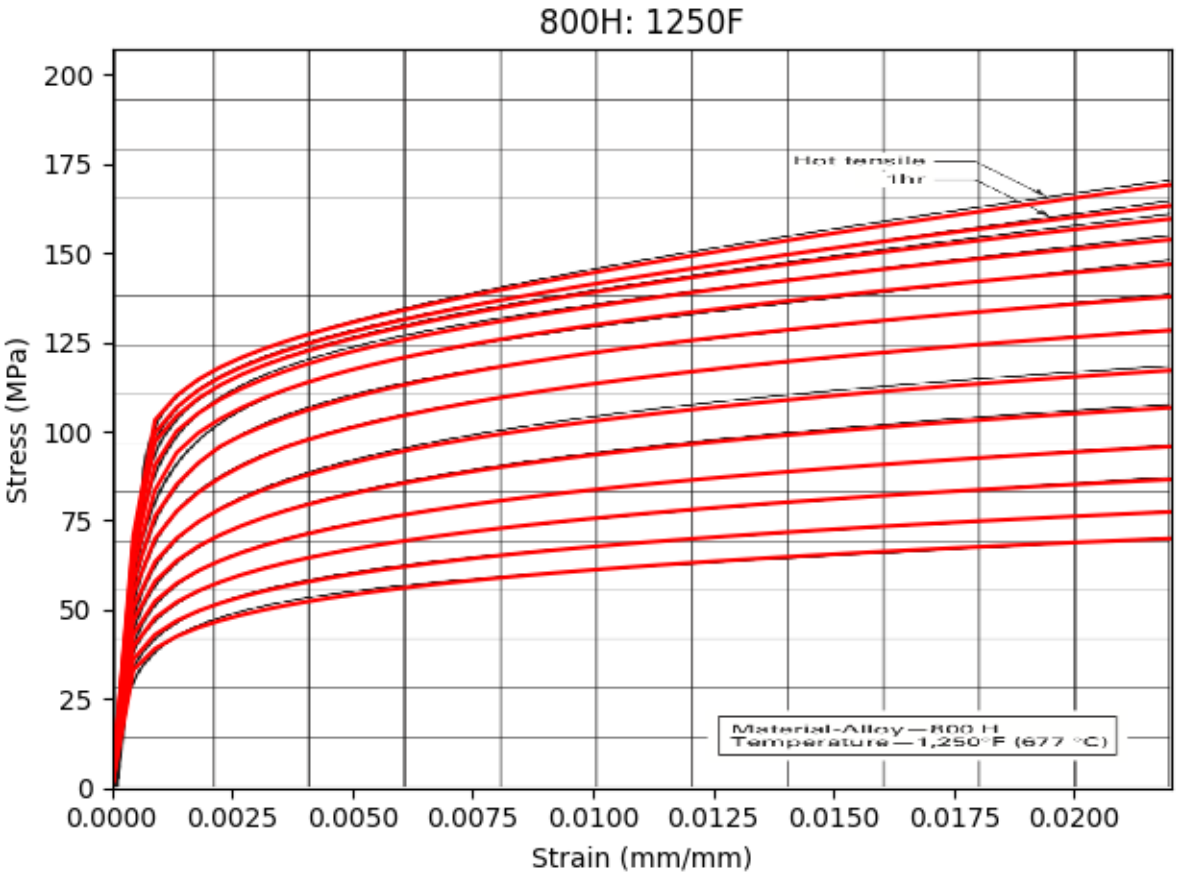

Fig. 36: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1250°F.

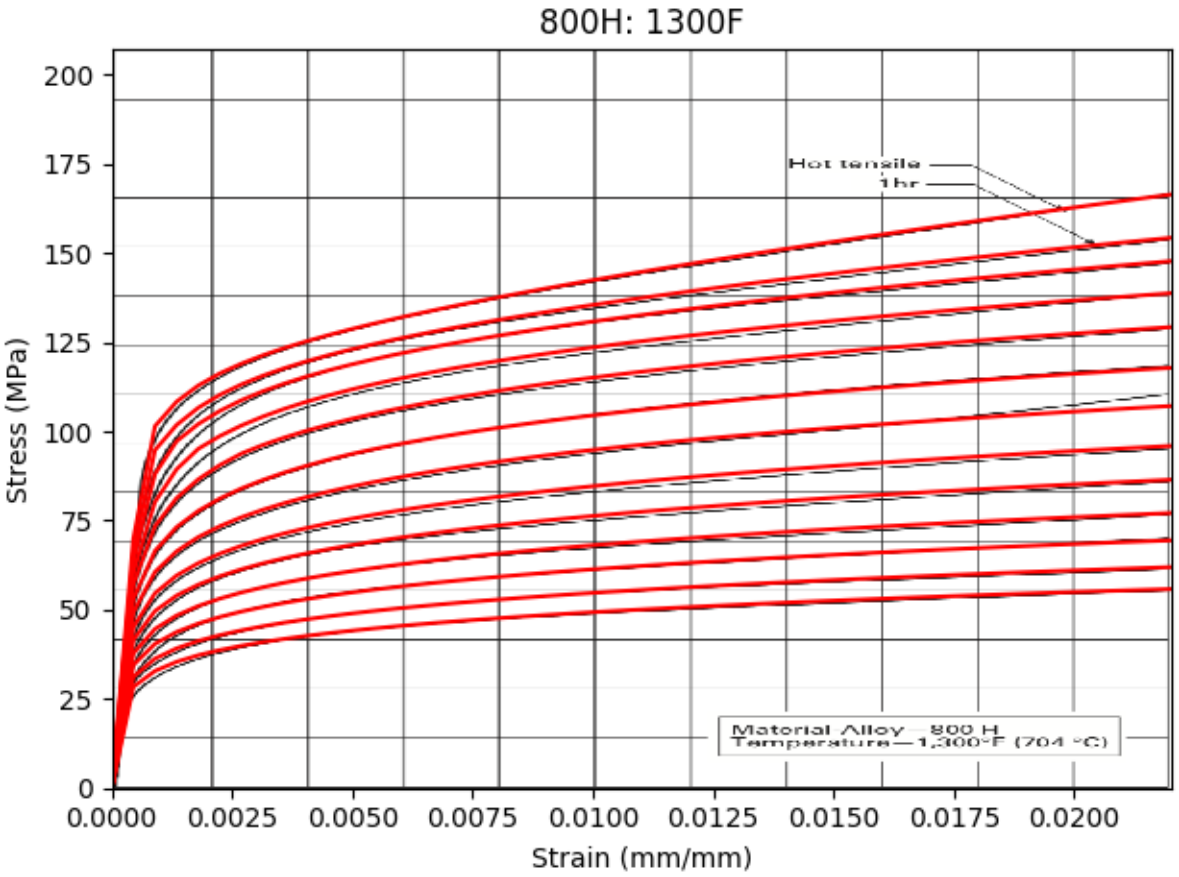

Fig. 37: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1300°F.

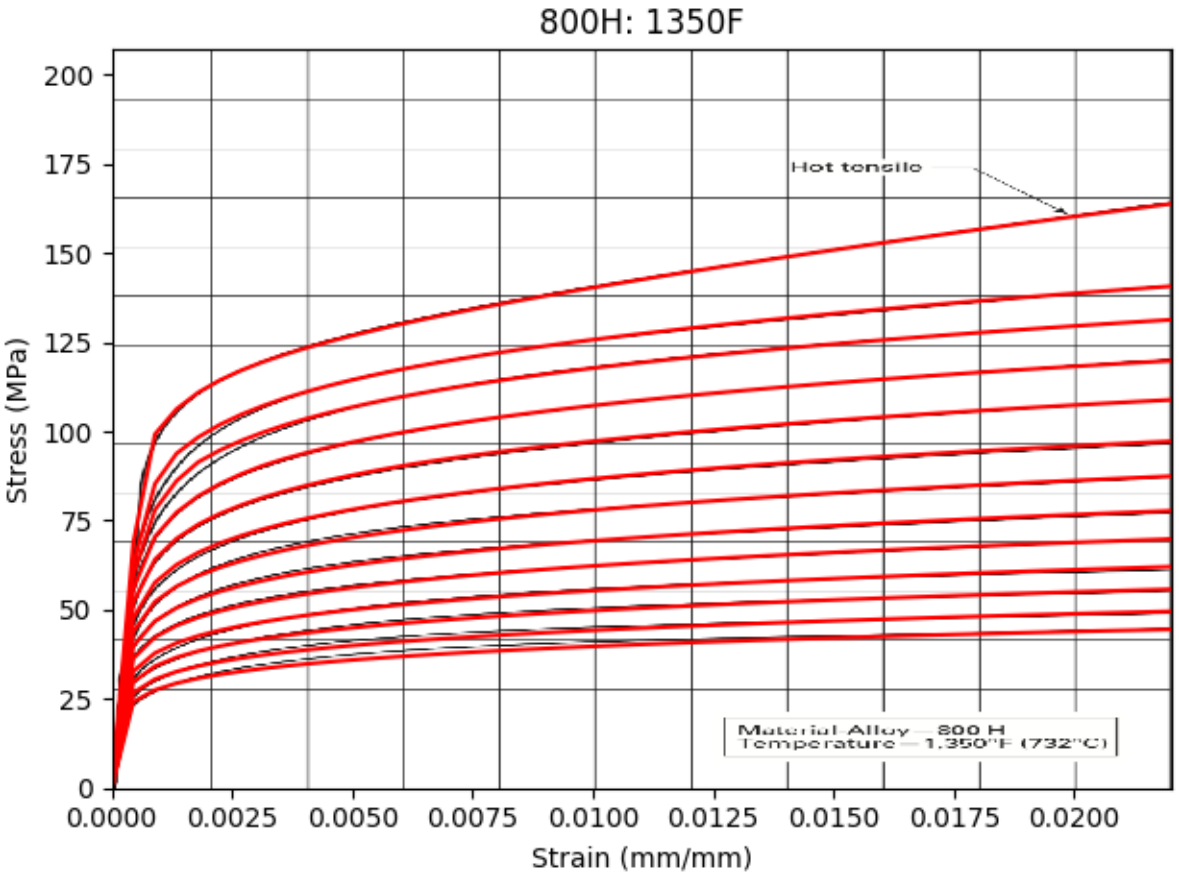

Fig. 38: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1350°F.

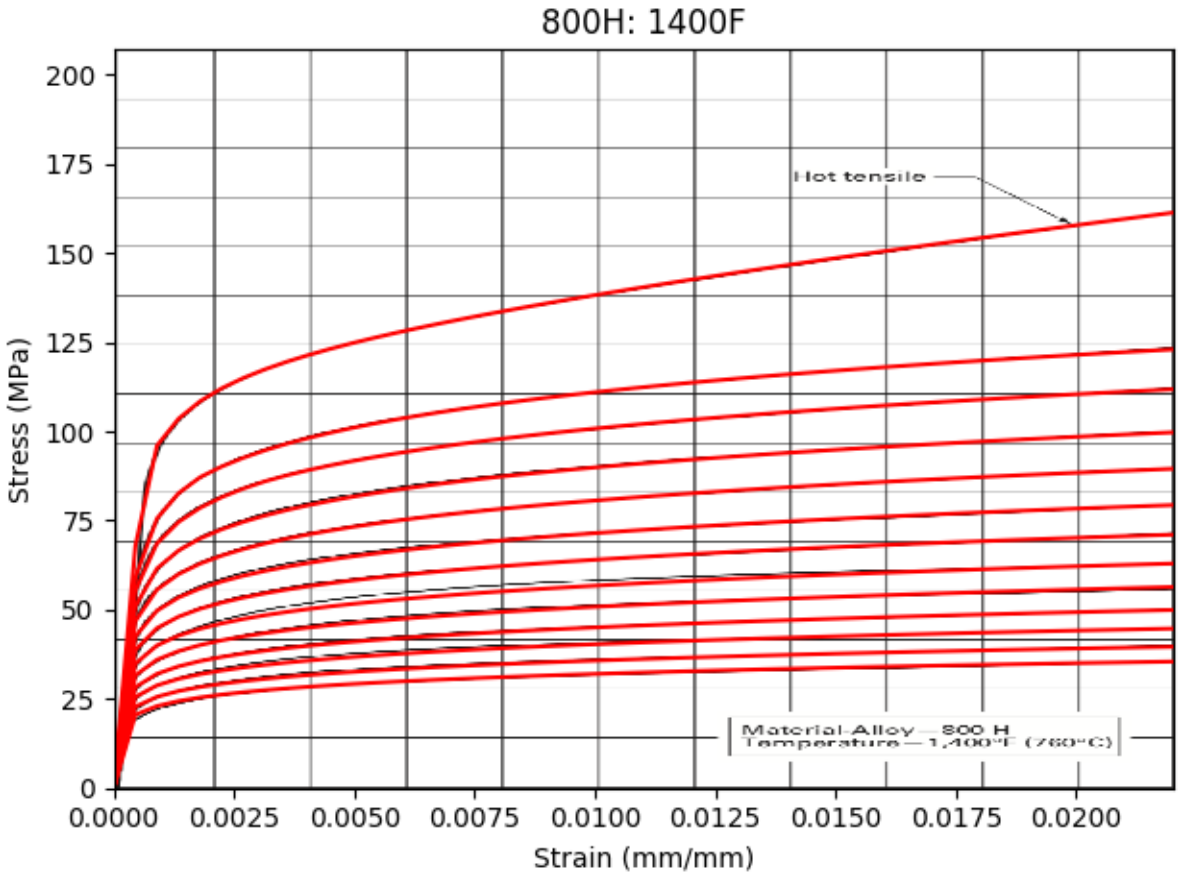

Fig. 39: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 1400°F.

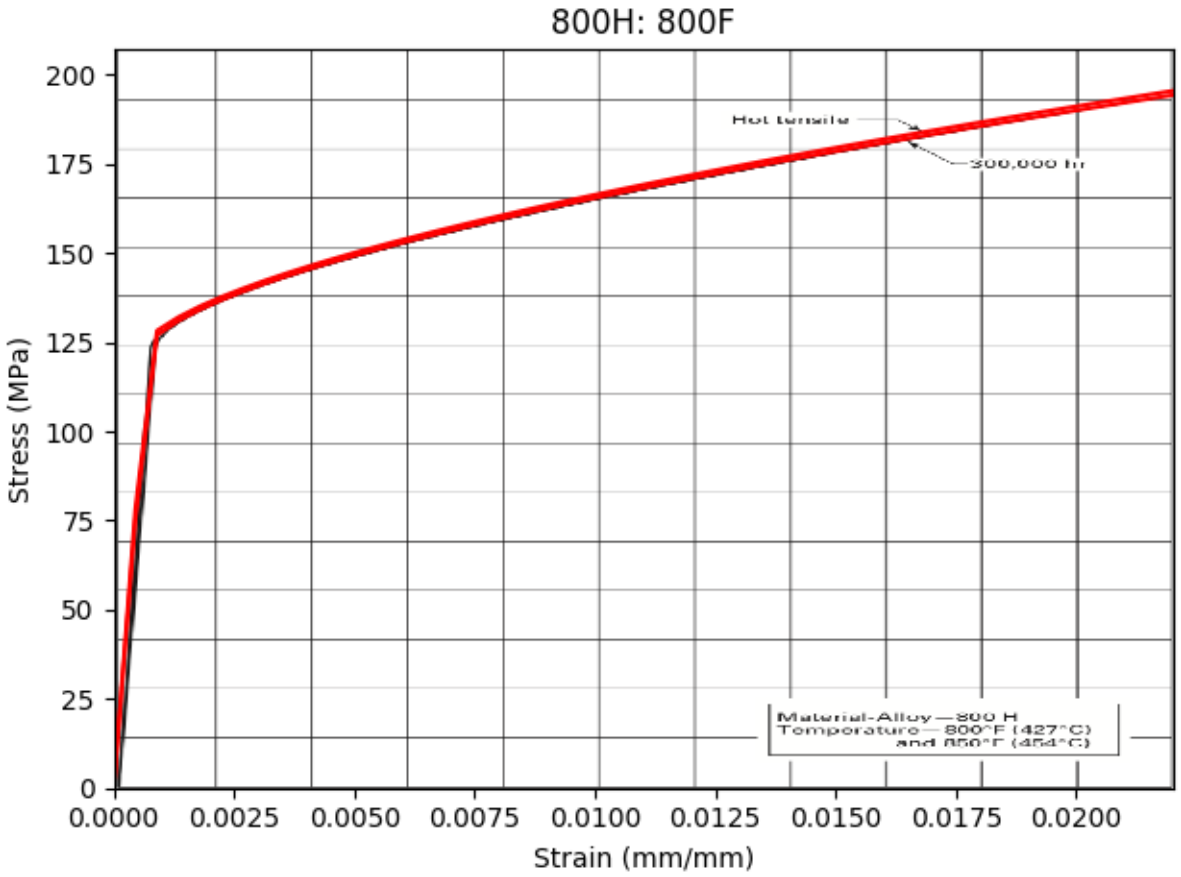

Fig. 40: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 800°F.

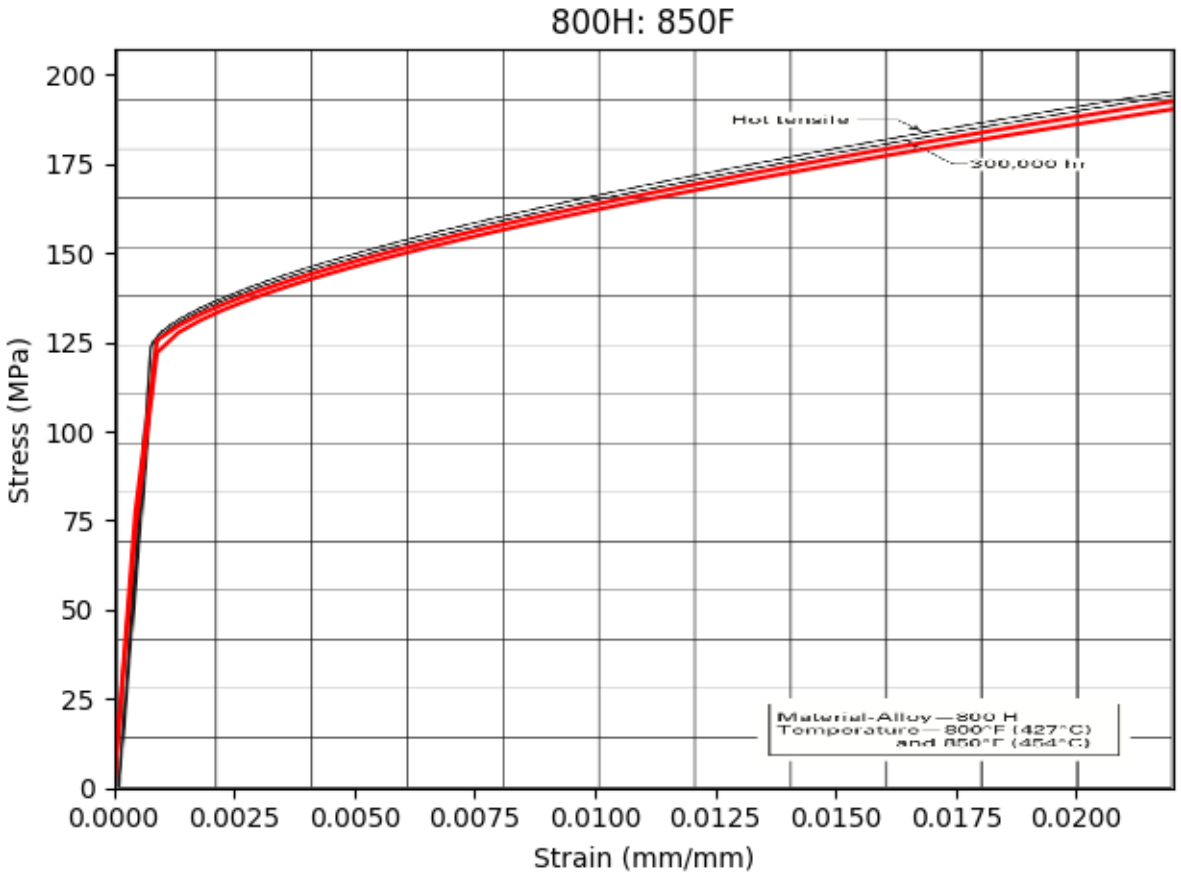

Fig. 41: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 850°F.

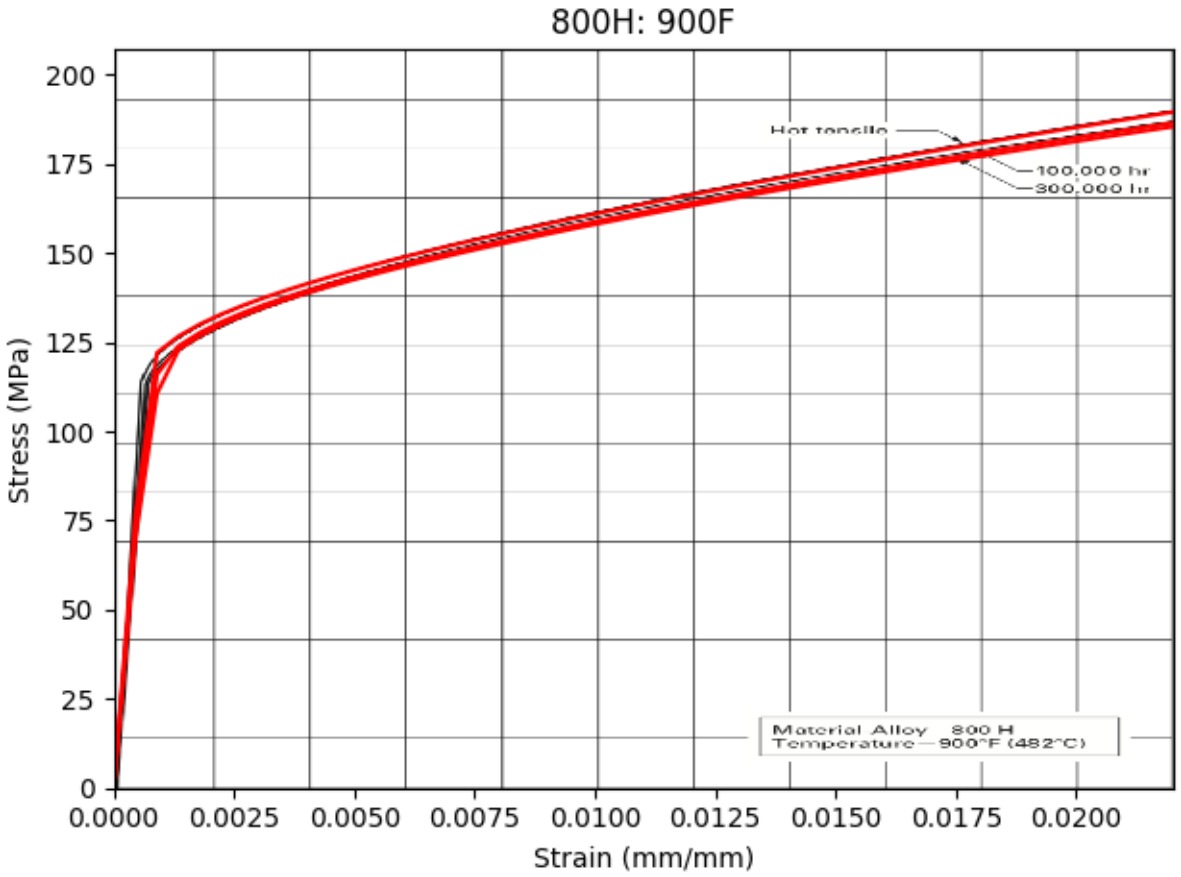

Fig. 42: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 900°F.

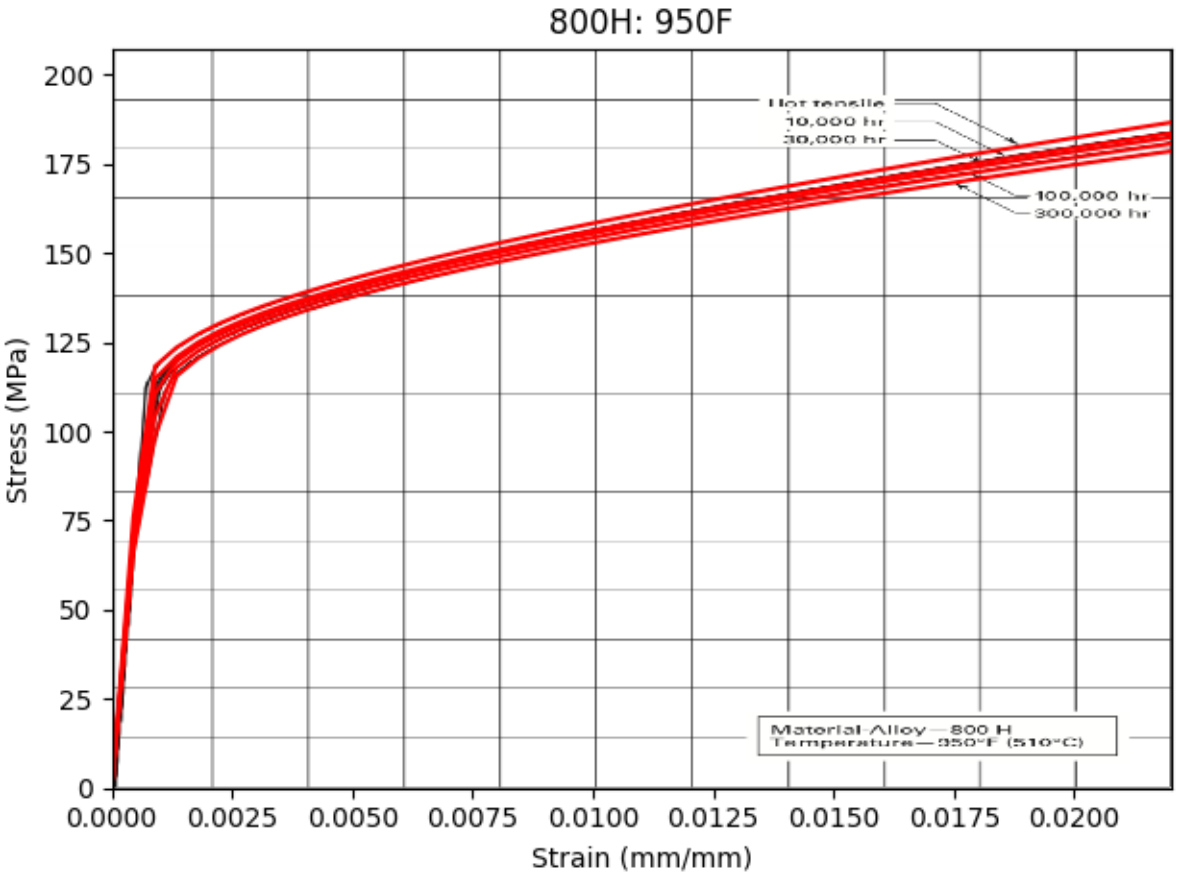

Fig. 43: Comparison between the current and implemented isochronous stress-strain curves for Alloy 800H at 950°F.

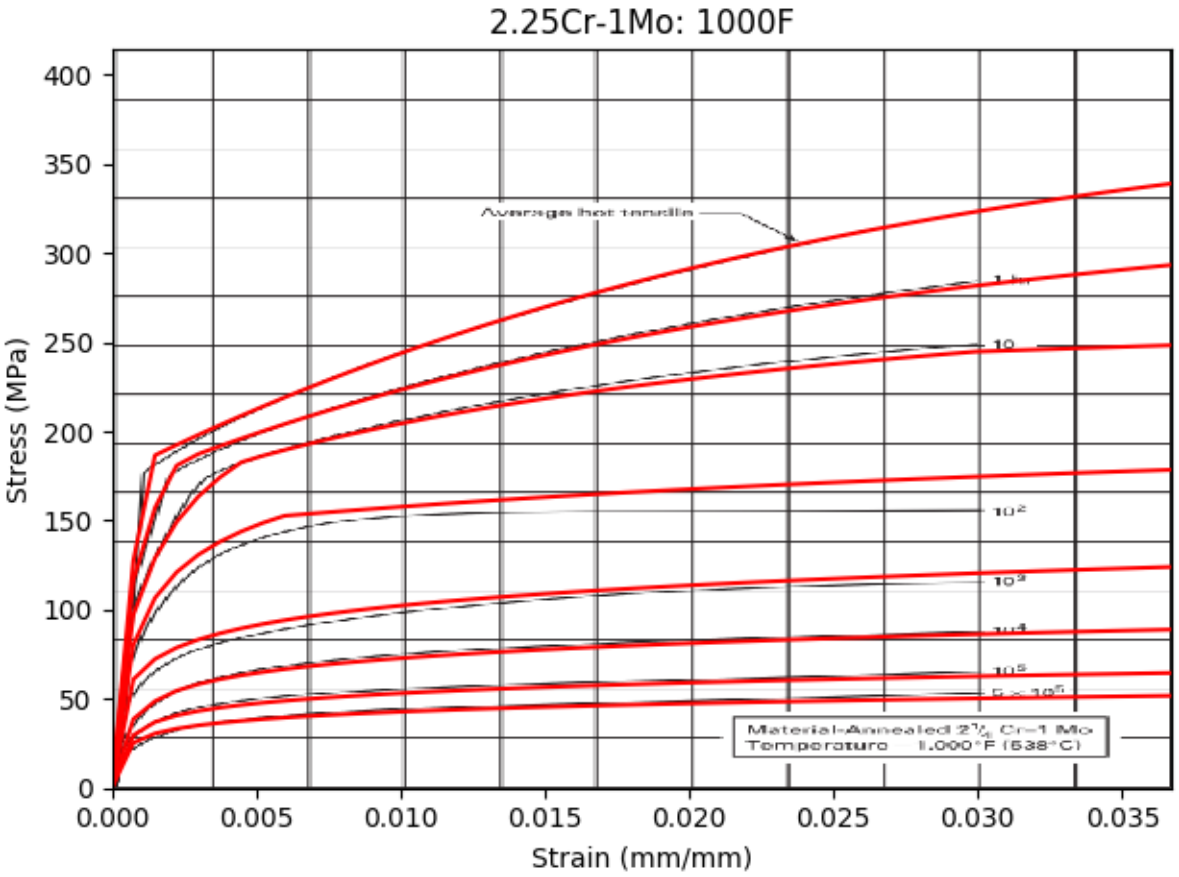

Fig. 44: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 1000°F.

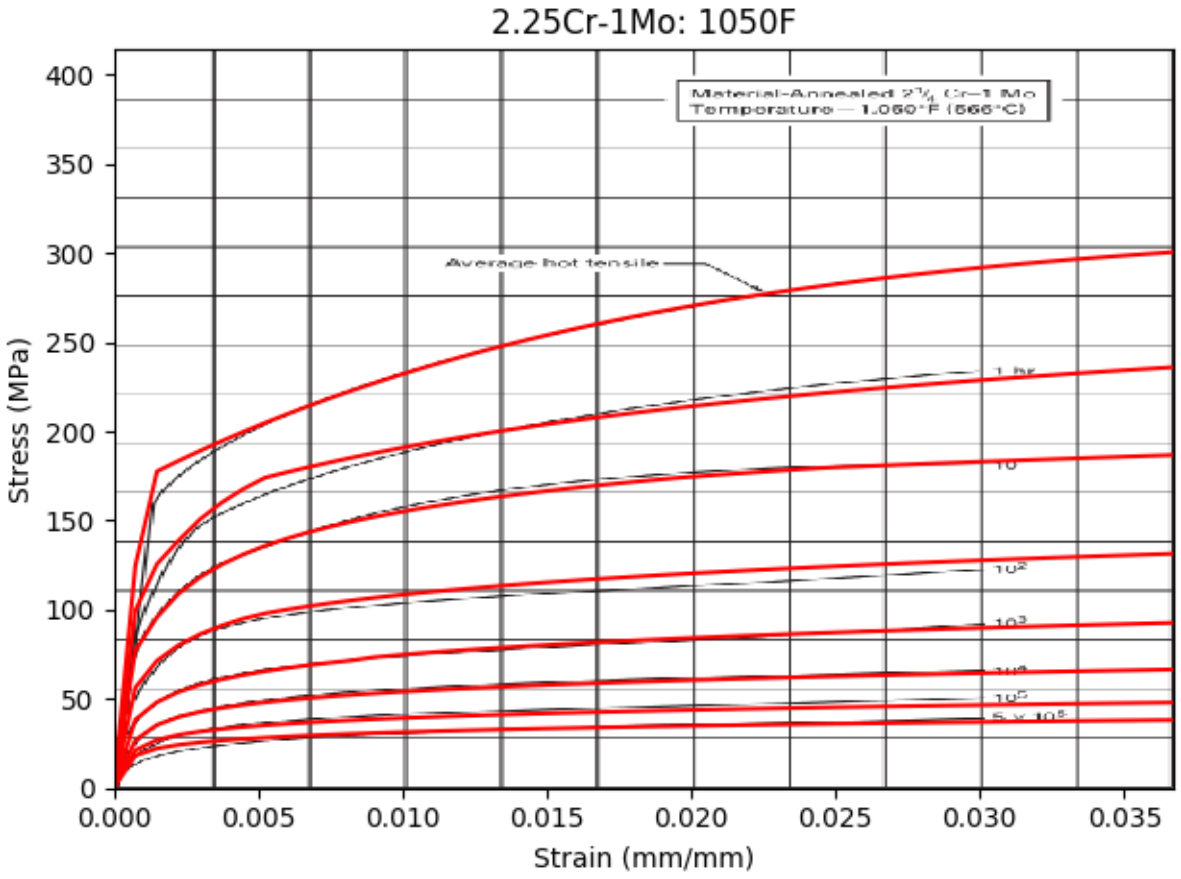

Fig. 45: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 1050°F.

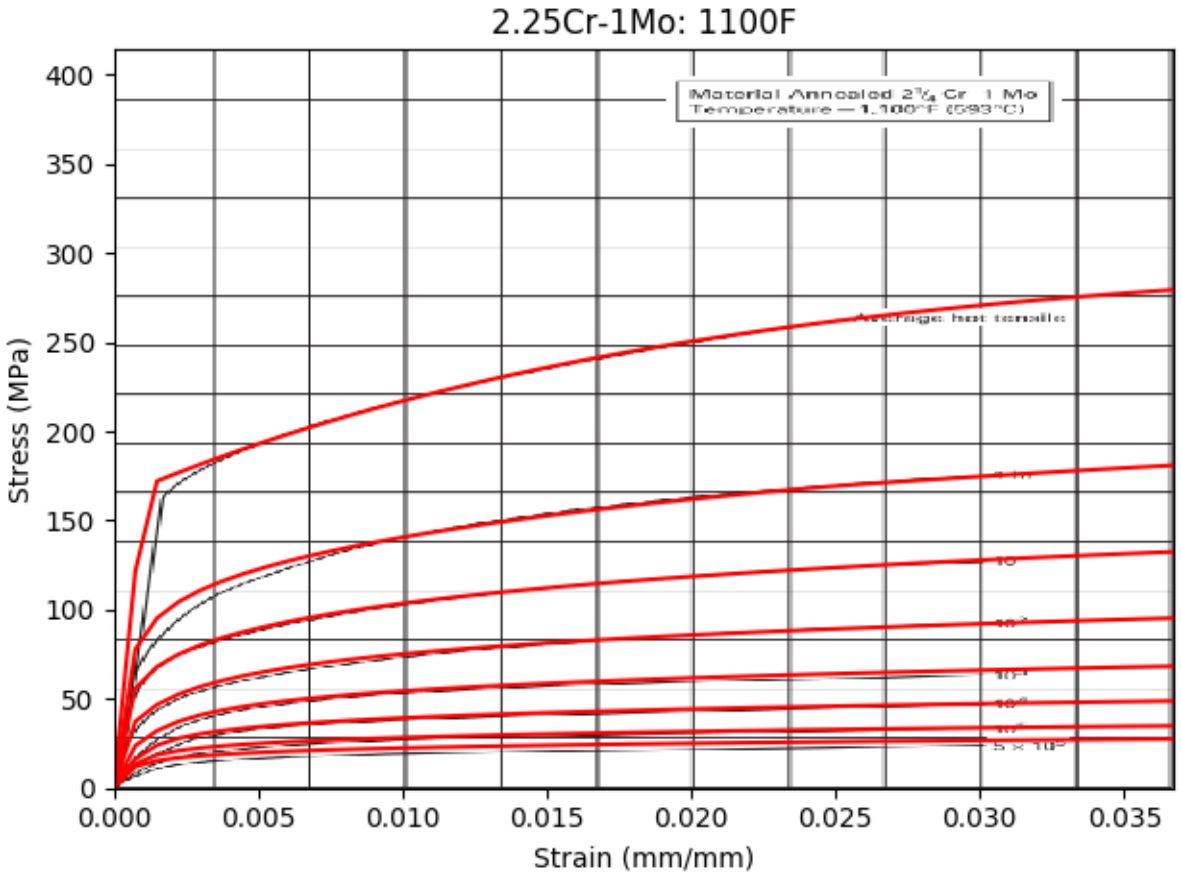

Fig. 46: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 1100°F.

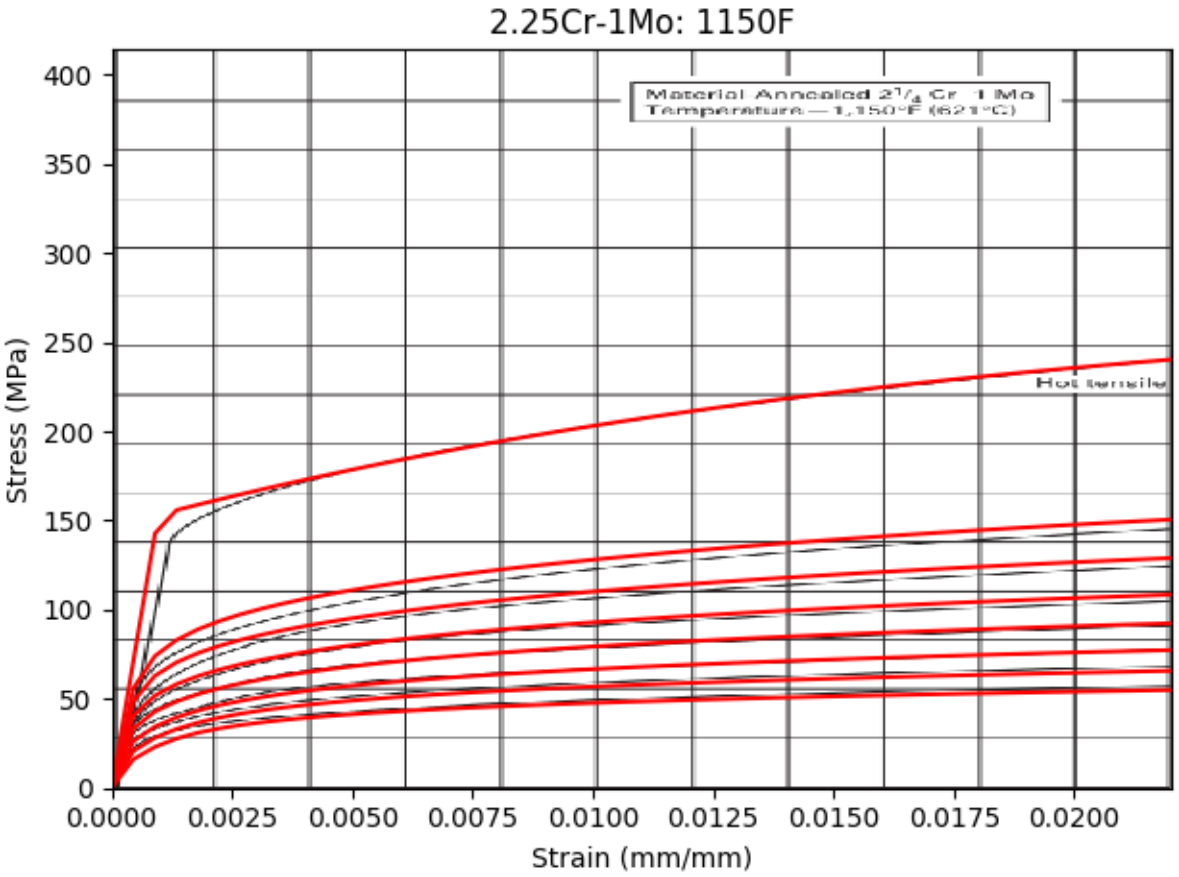

Fig. 47: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 1150°F.

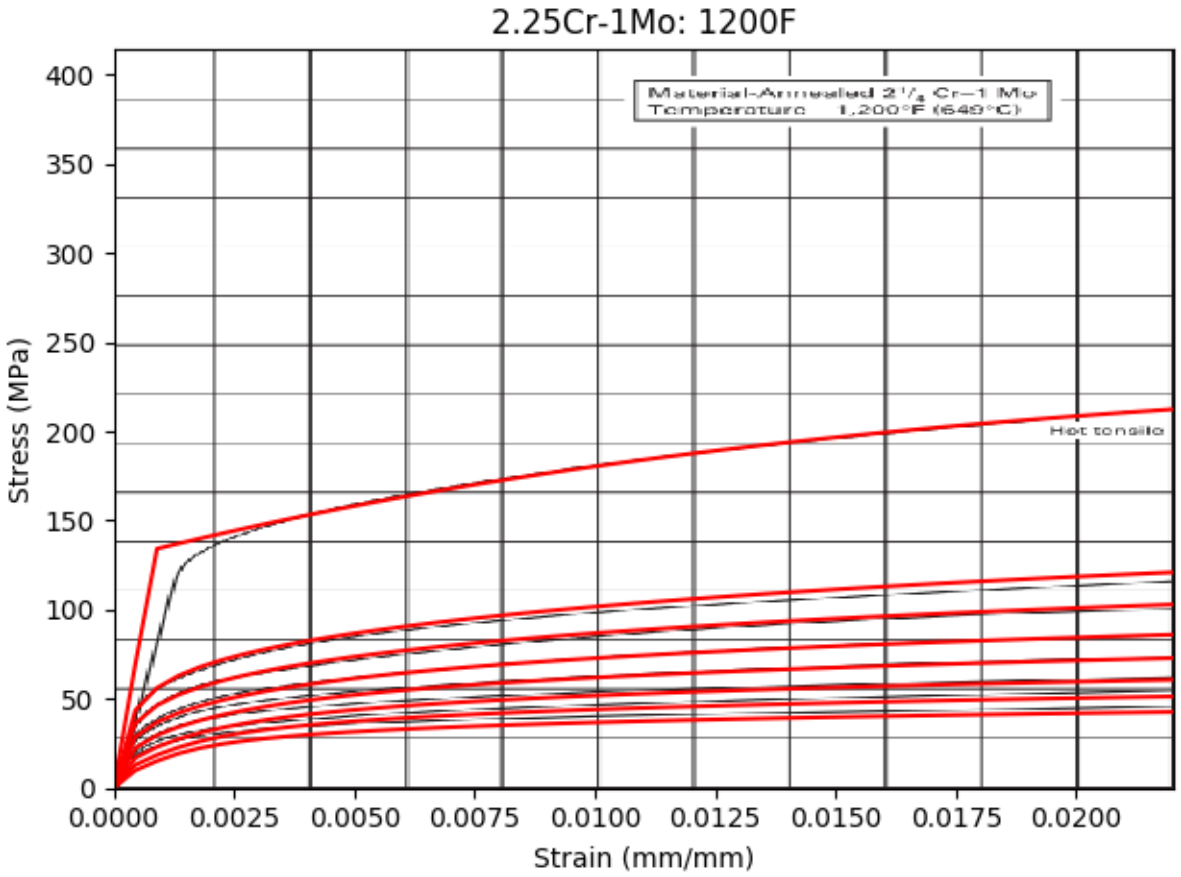

Fig. 48: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 1200°F.

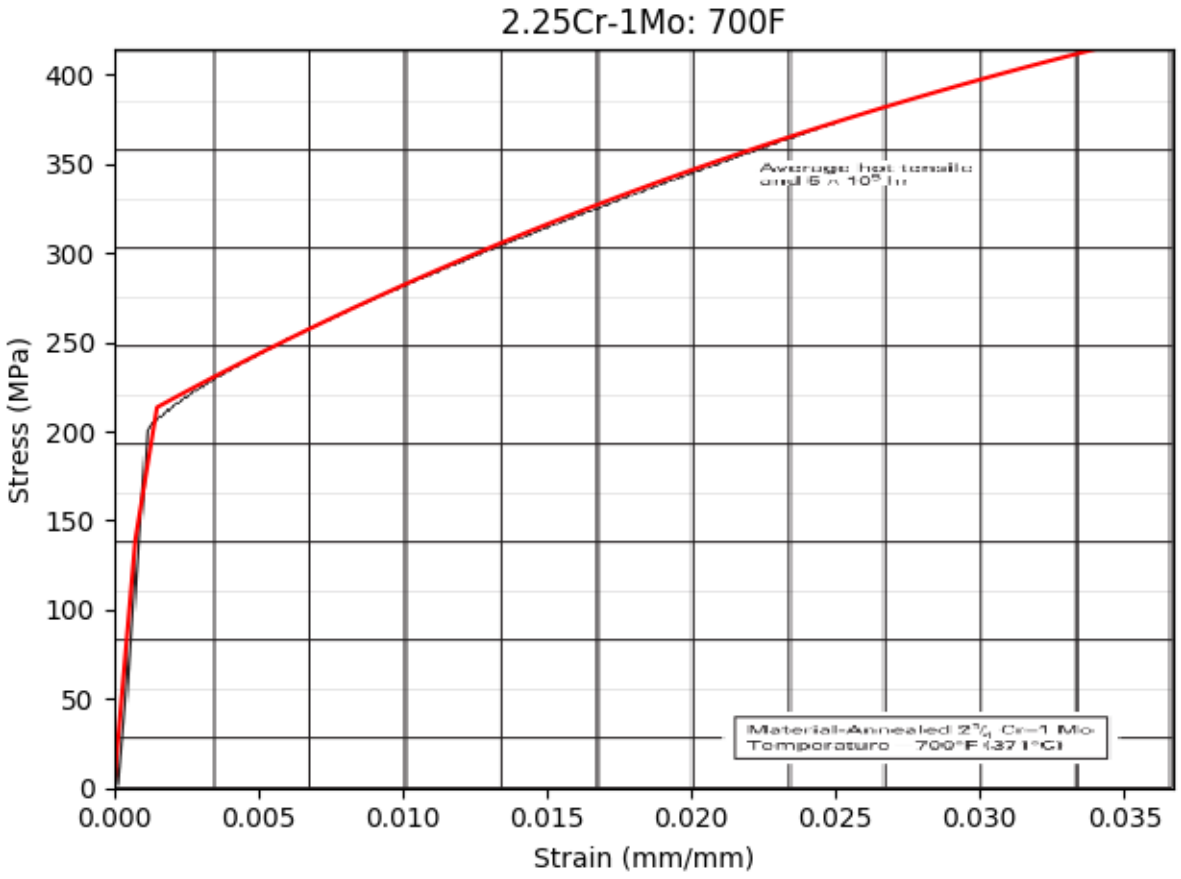

Fig. 49: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 700°F.

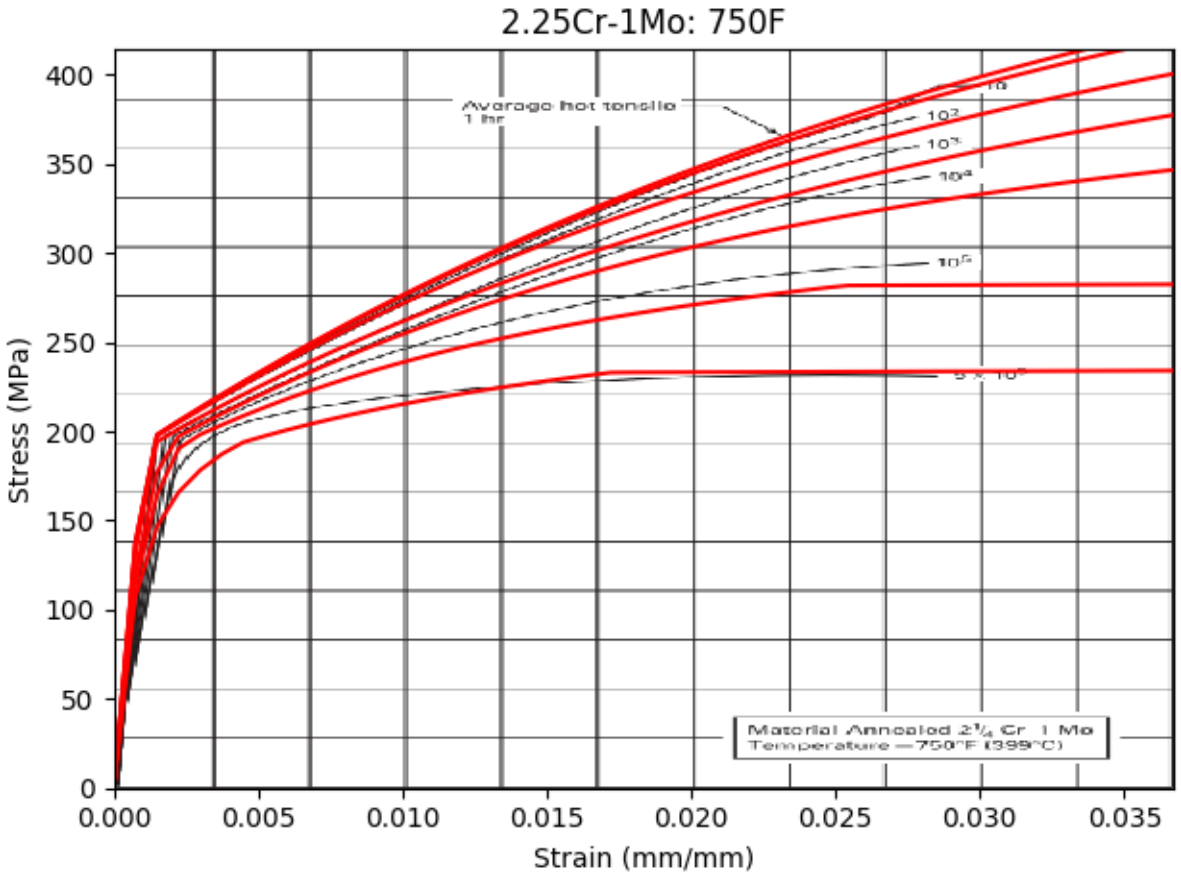

Fig. 50: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 750°F.

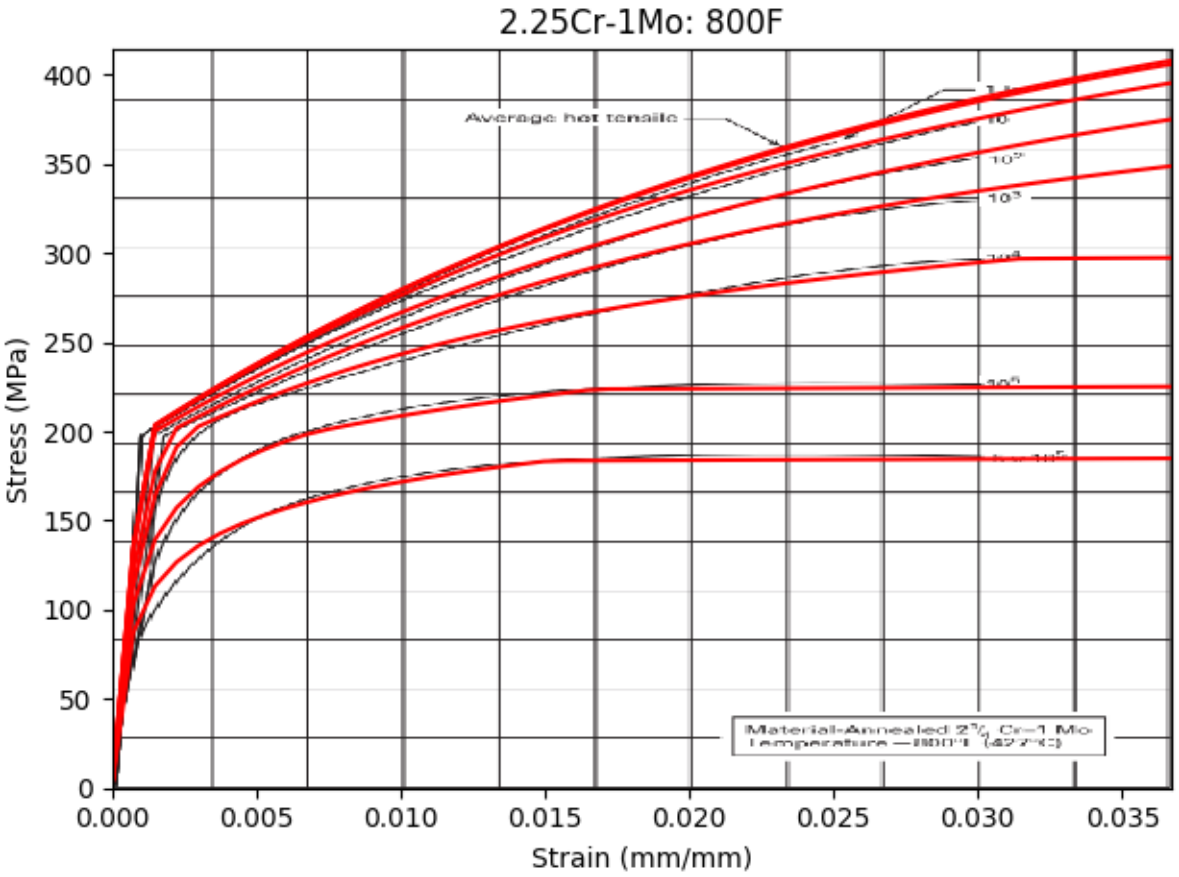

Fig. 51: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 800°F.

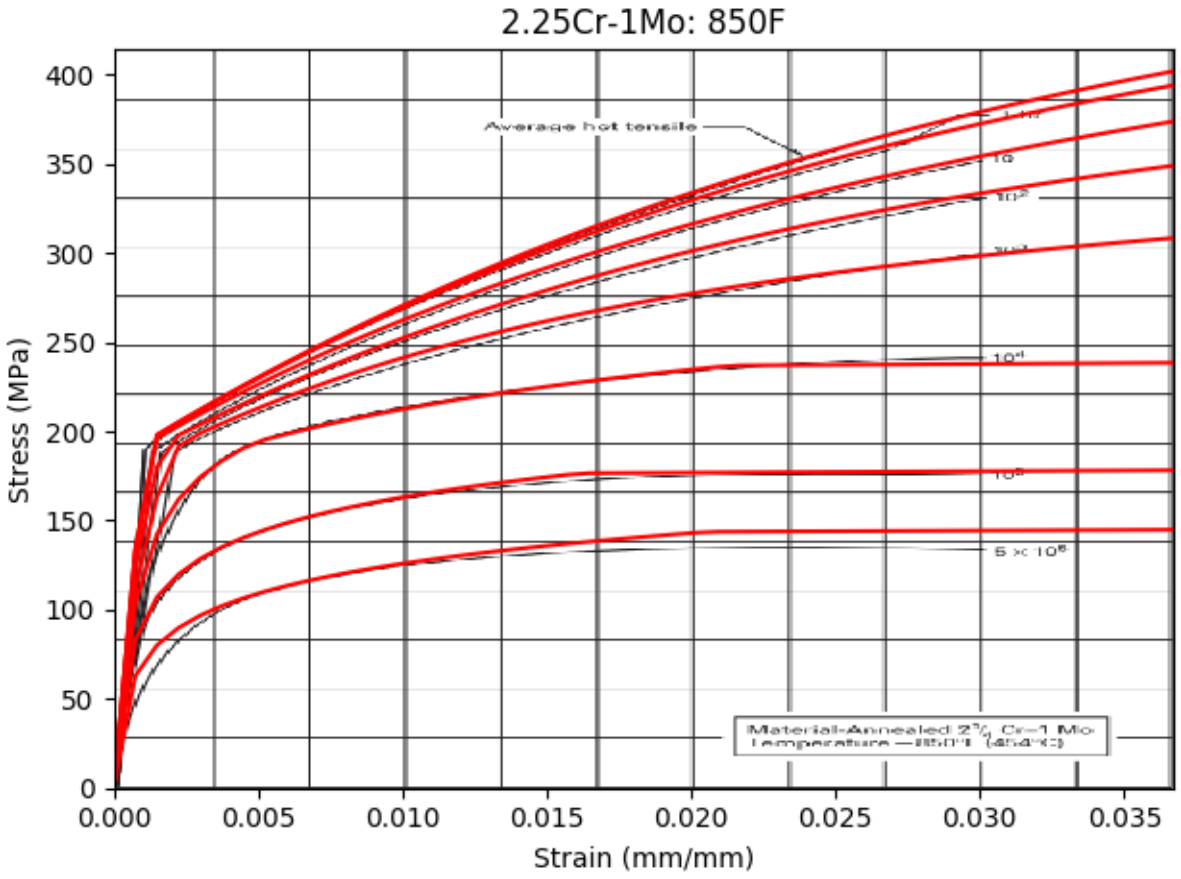

Fig. 52: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 850°F.

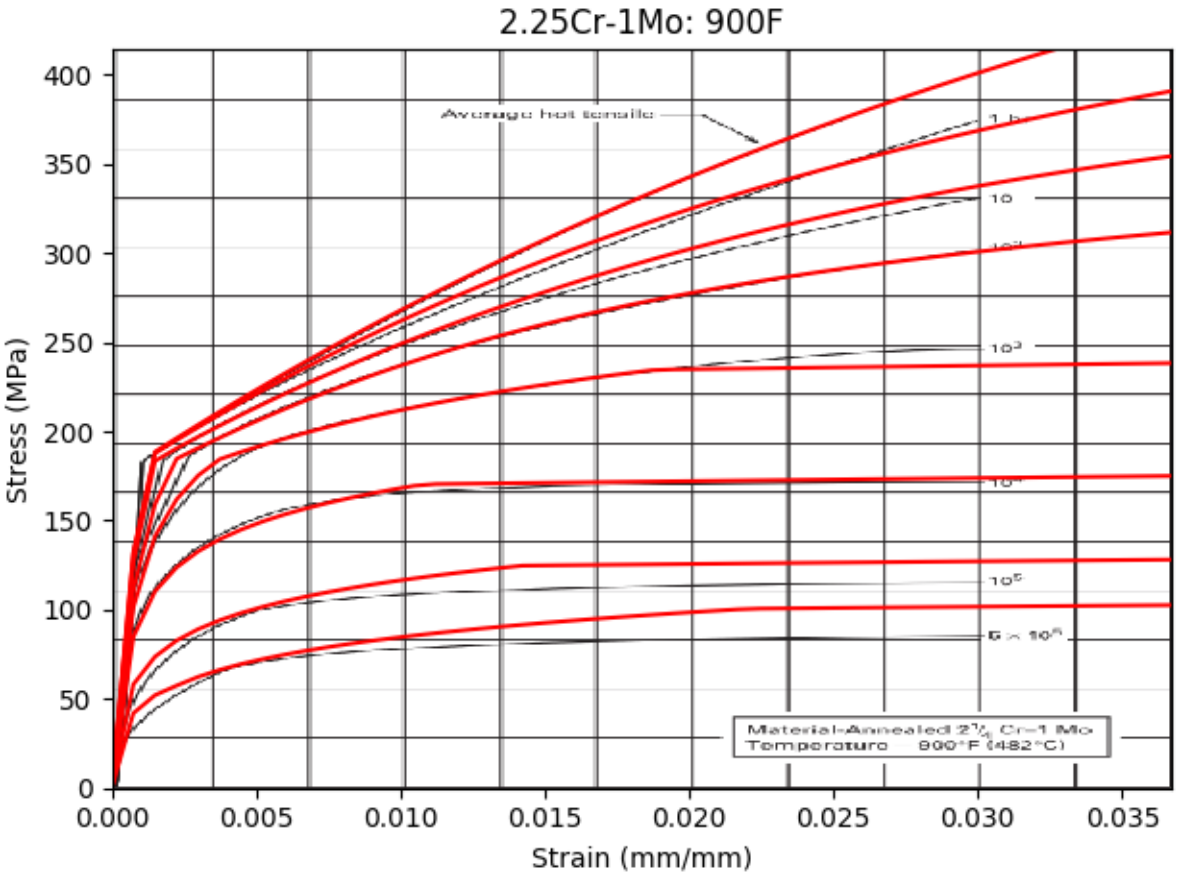

Fig. 53: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 900°F.

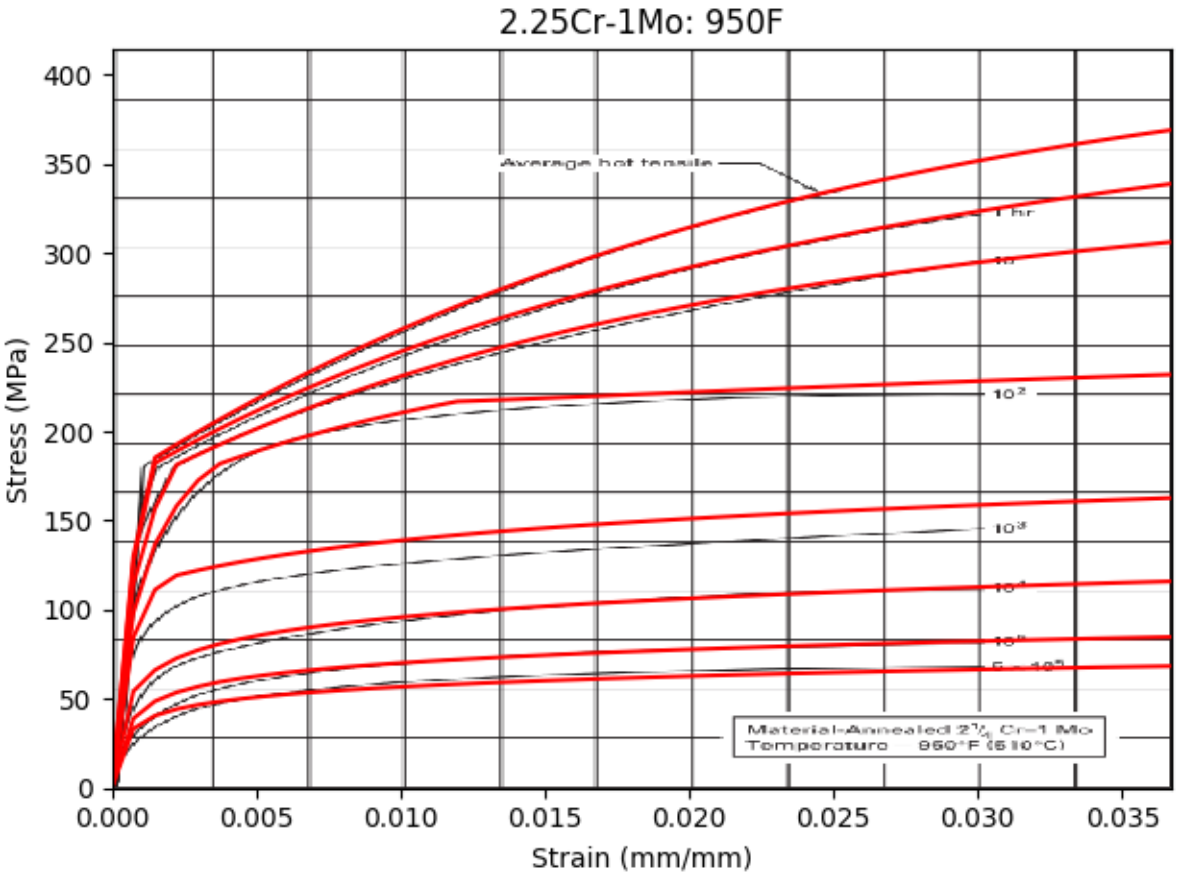

Fig. 54: Comparison between the current and implemented isochronous stress-strain curves for 2.25Cr-1Mo at 950°F.

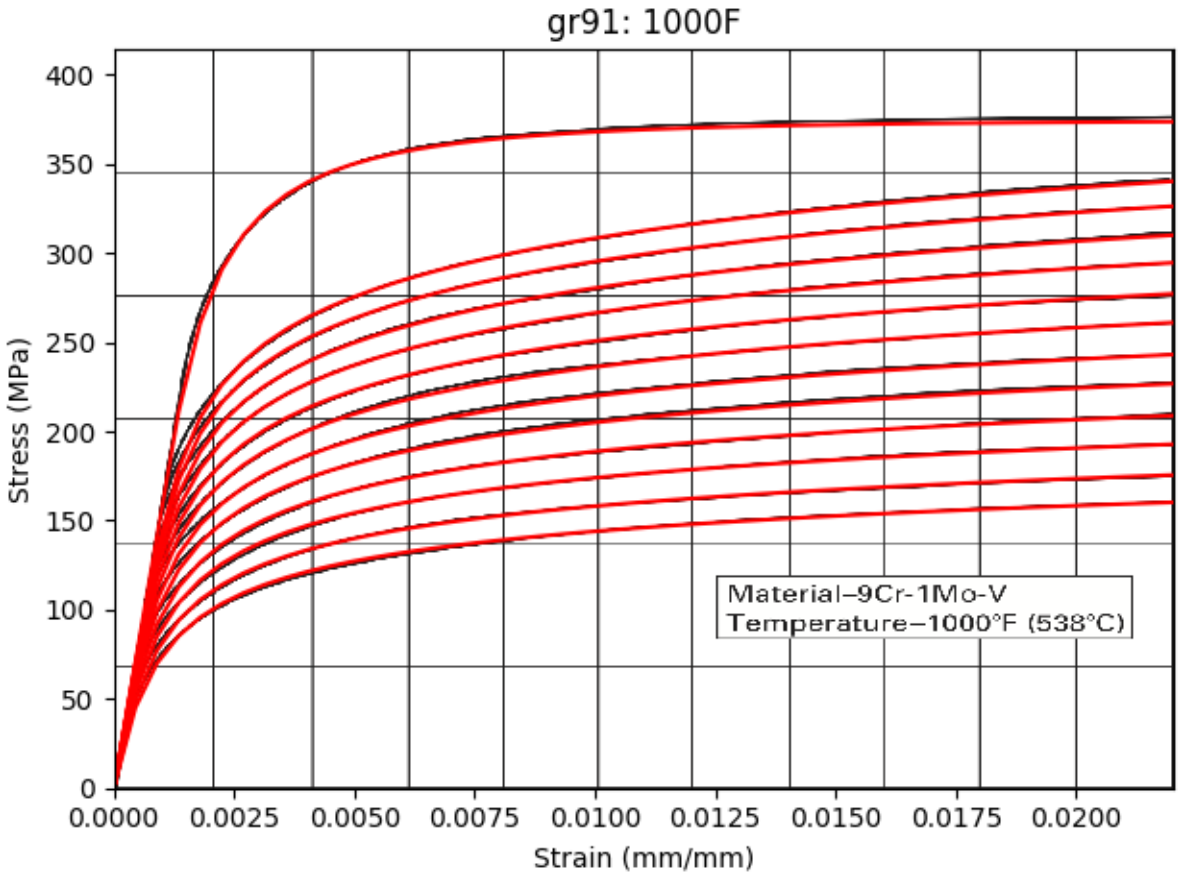

Fig. 55: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 1000°F.

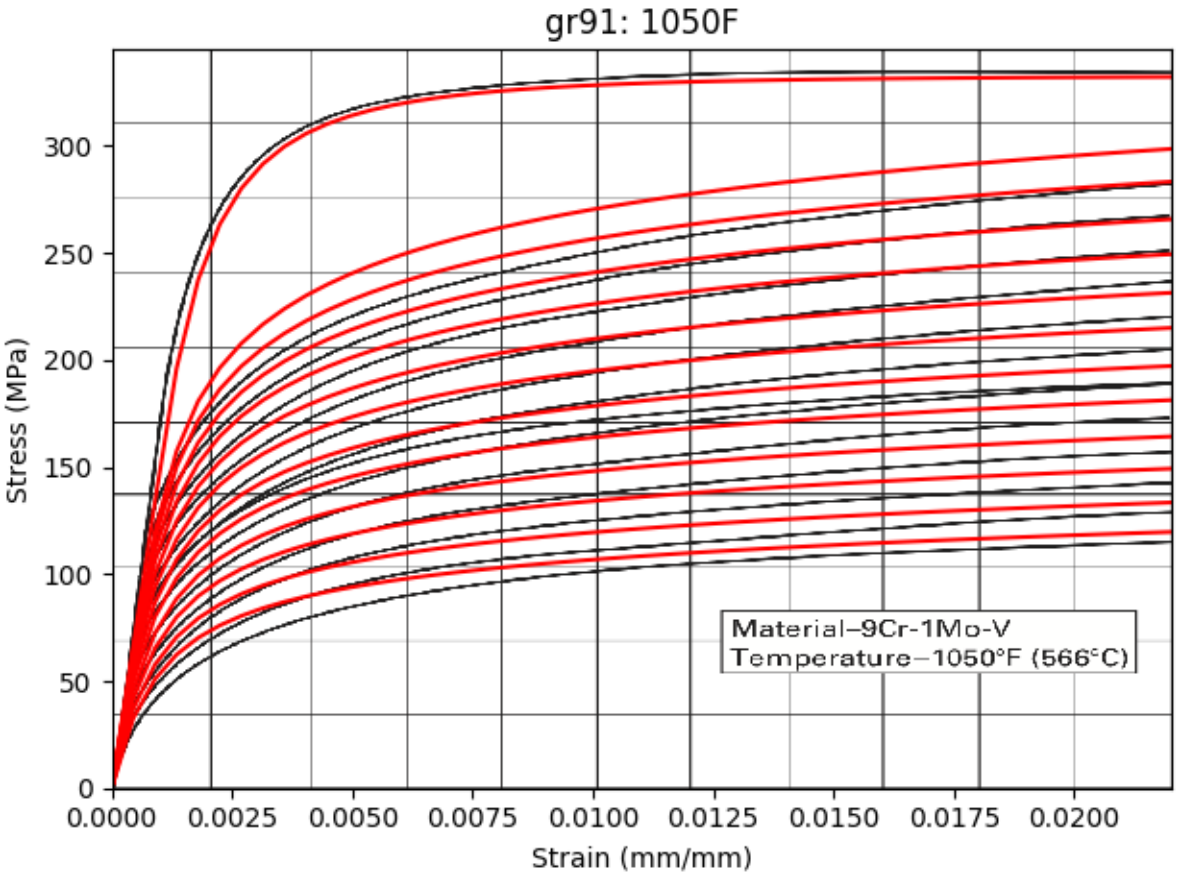

Fig. 56: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 1050°F.

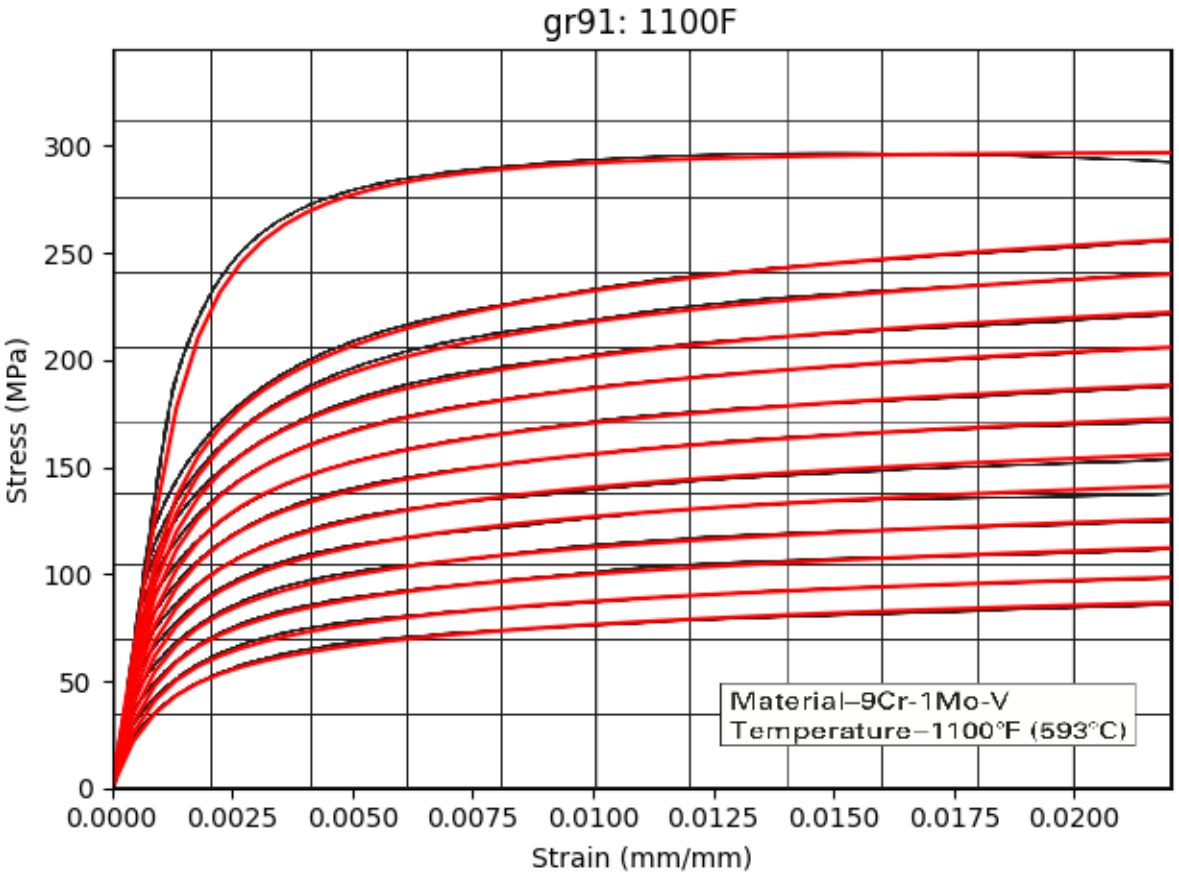

Fig. 57: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 1100°F.

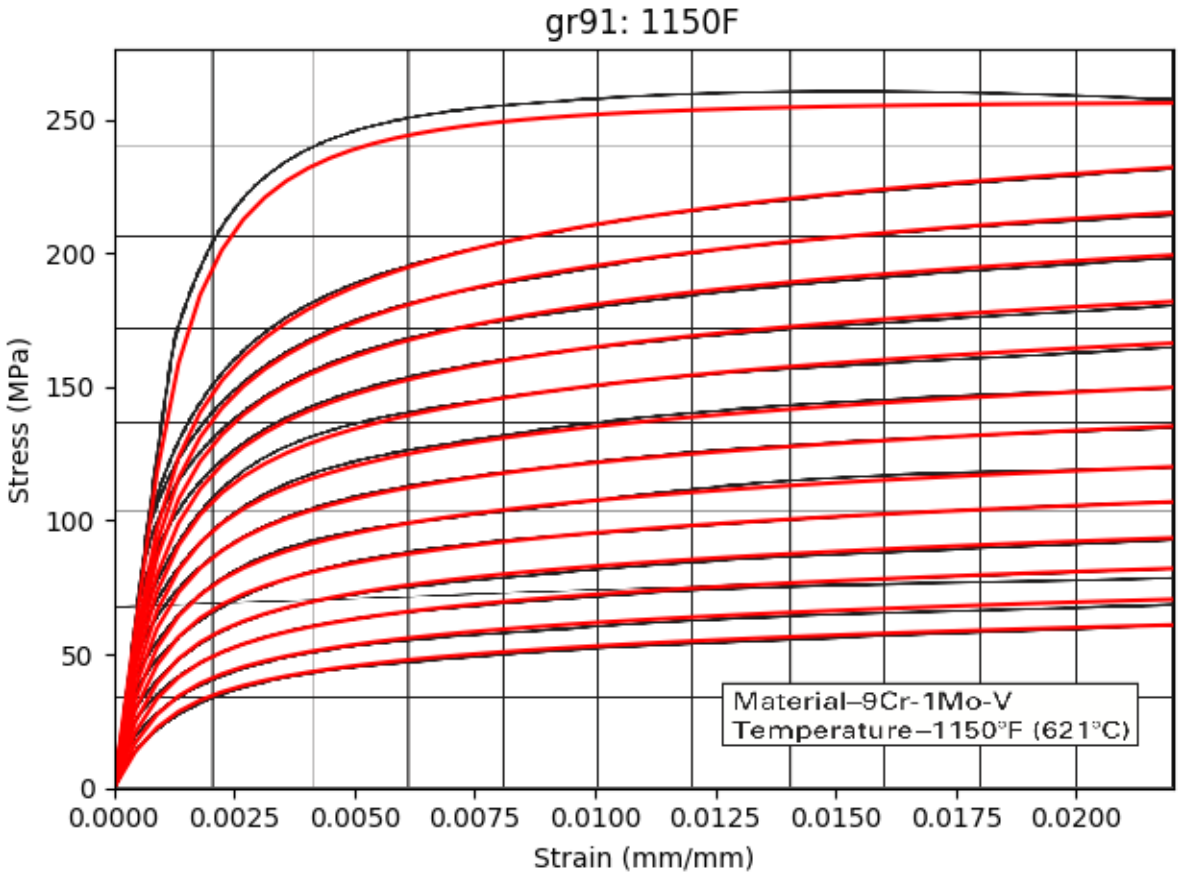

Fig. 58: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 1150°F.

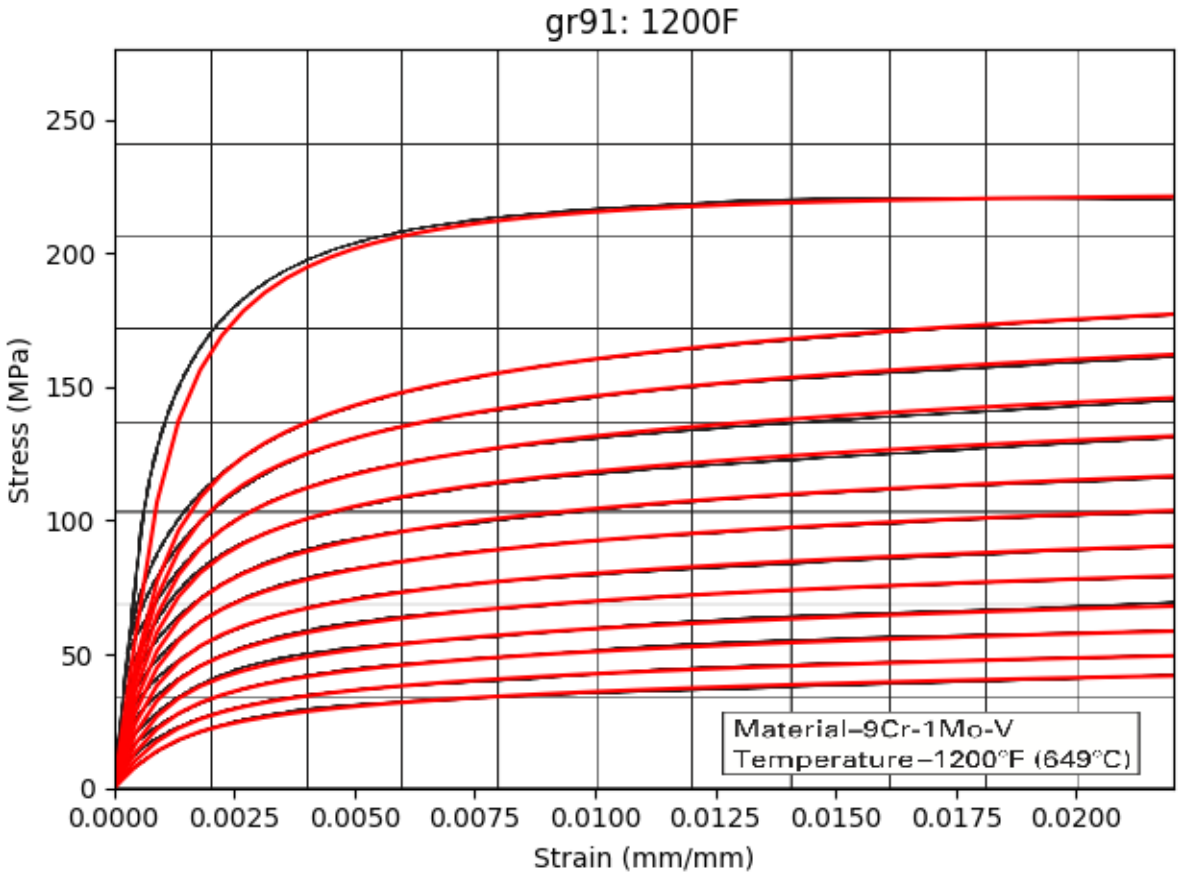

Fig. 59: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 1200°F.

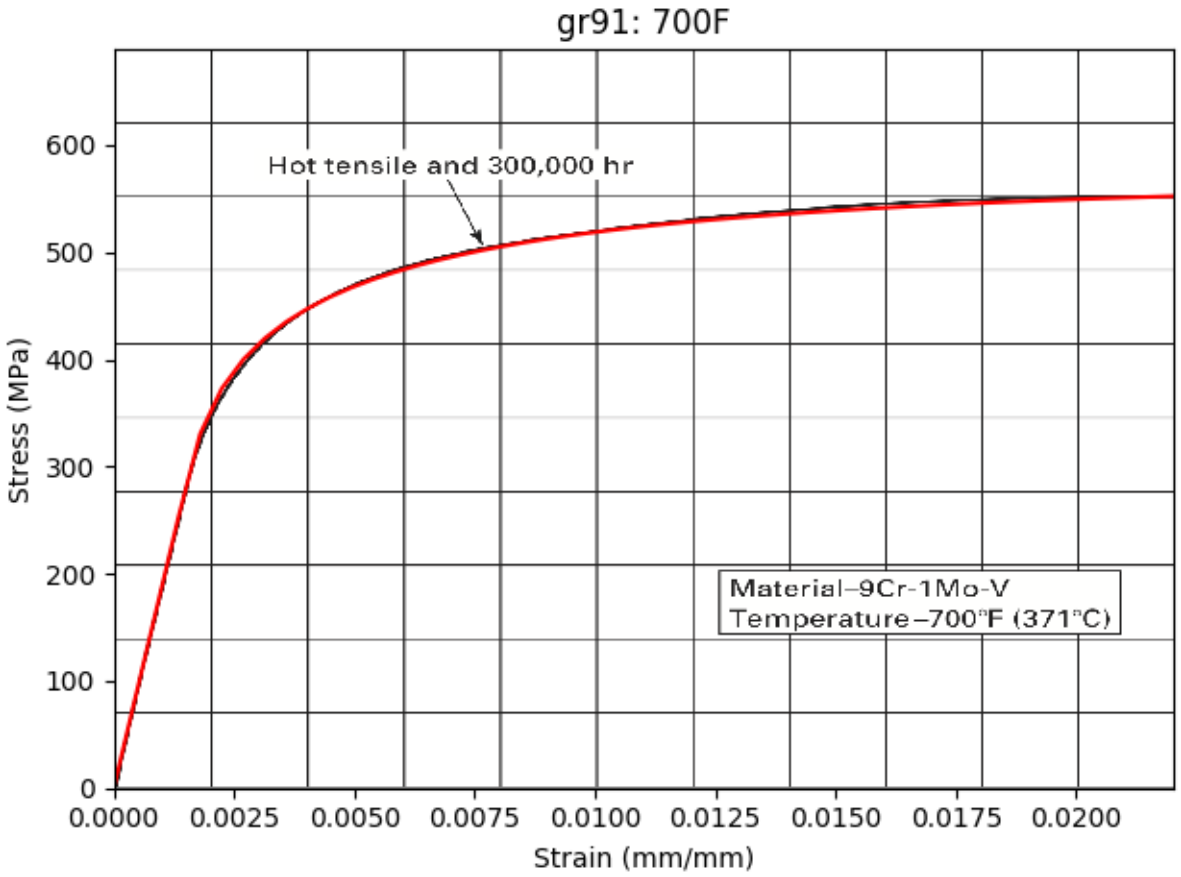

Fig. 60: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 700°F.

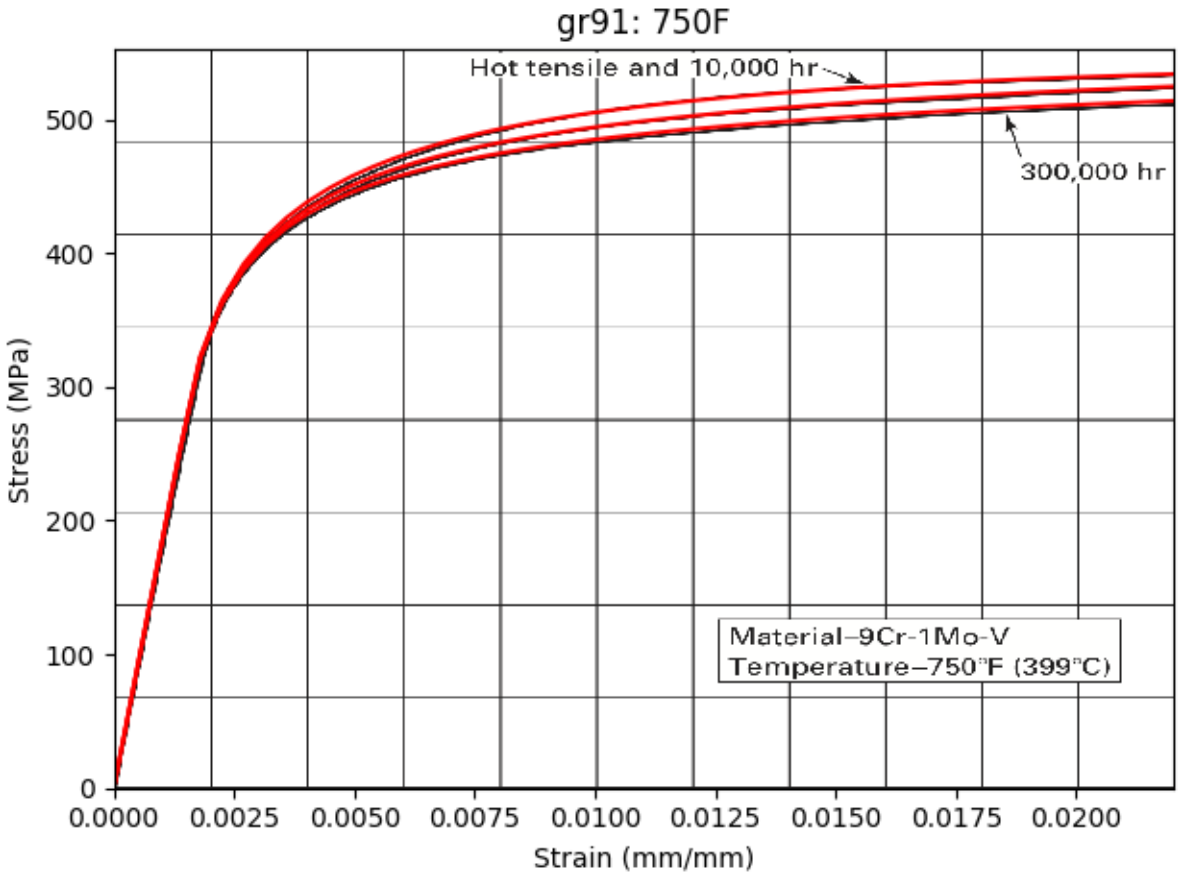

Fig. 61: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 750°F.

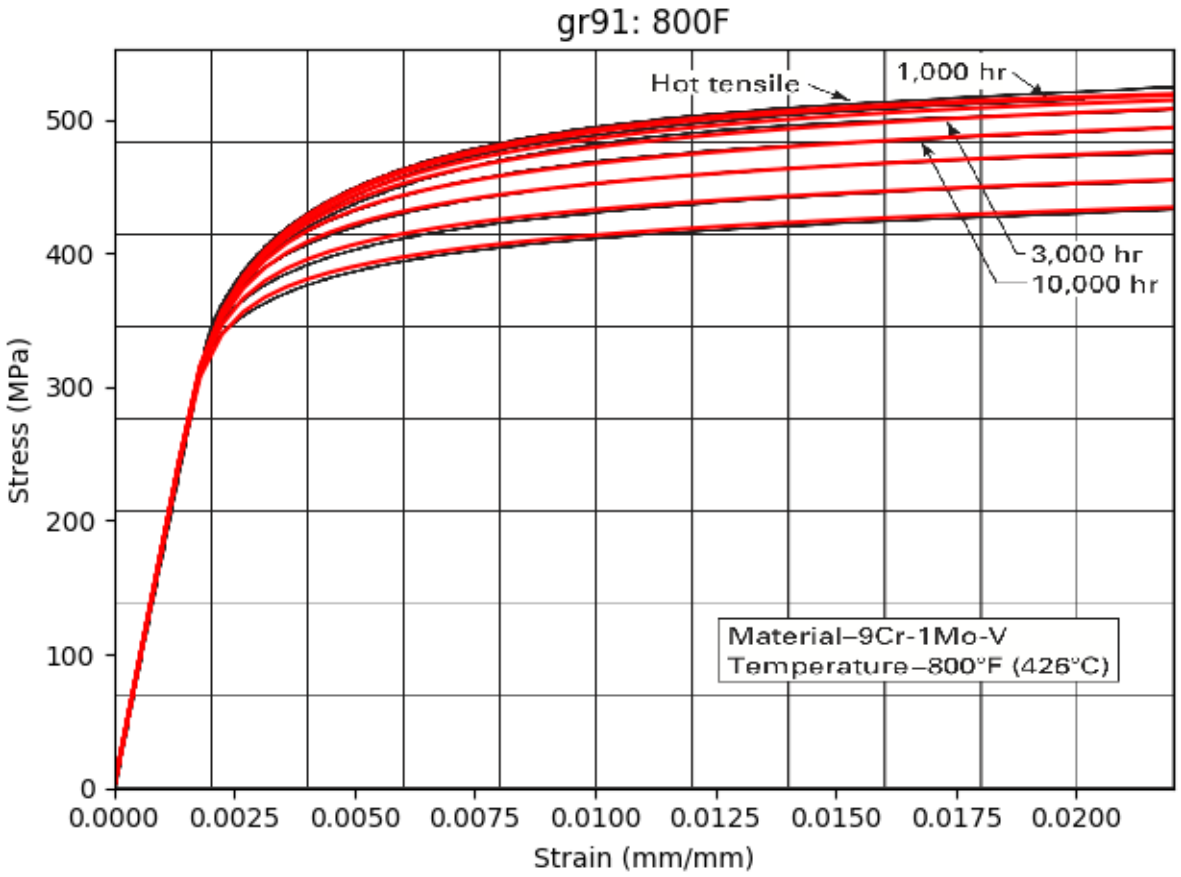

Fig. 62: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 800°F.

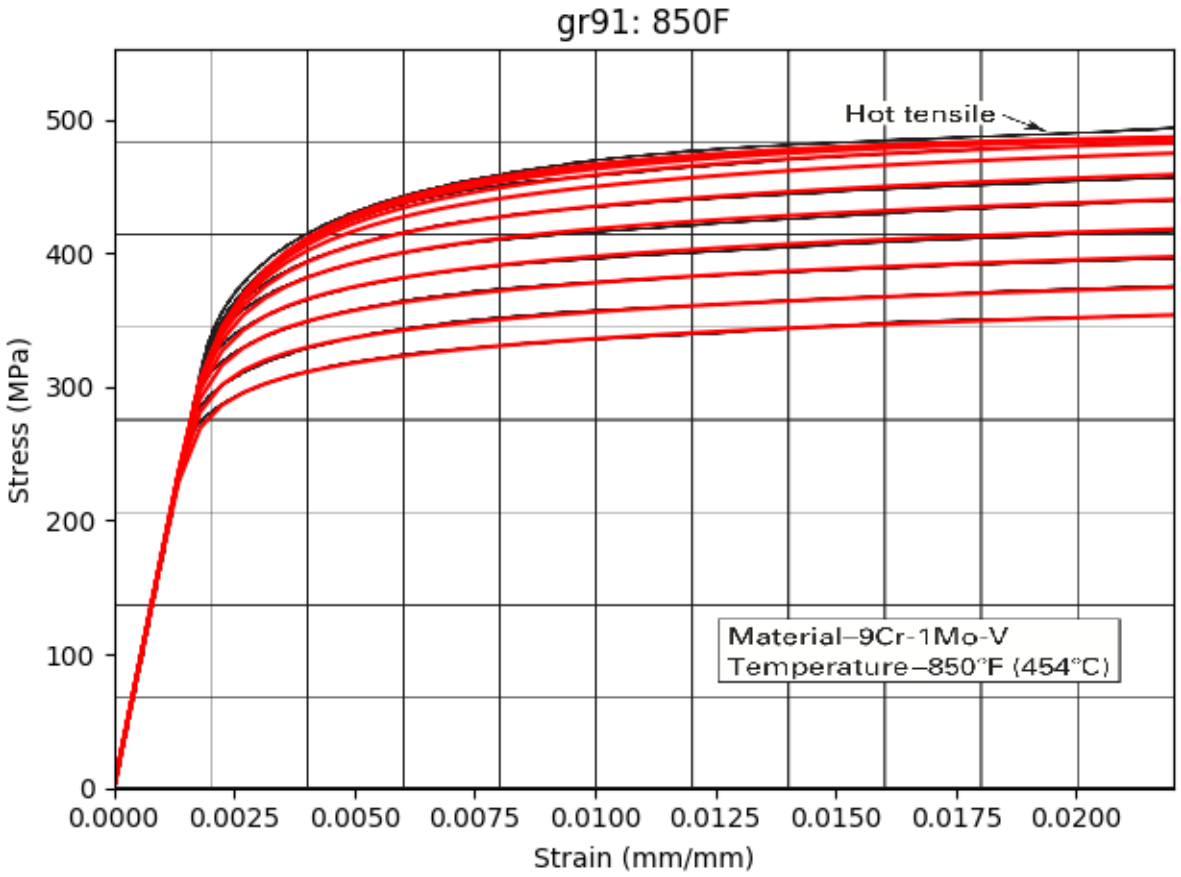

Fig. 63: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 850°F.

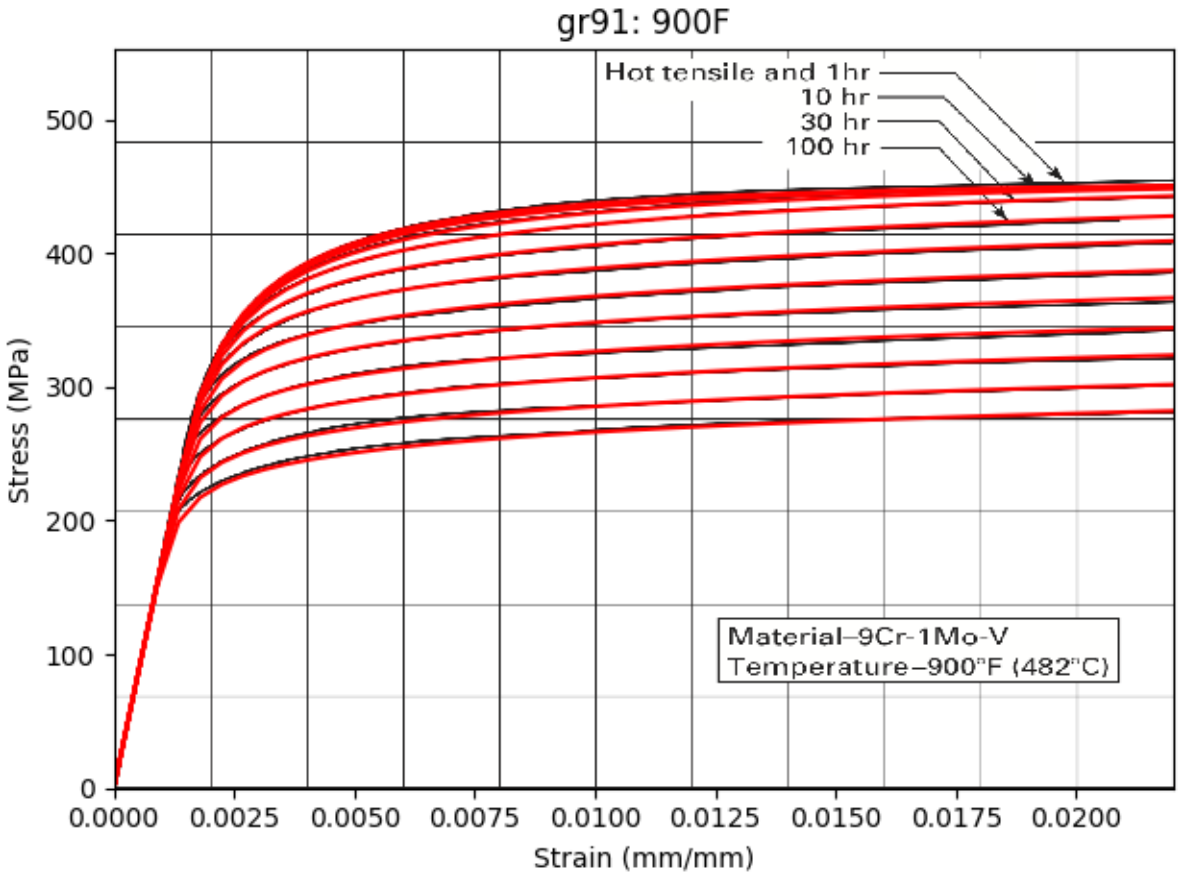

Fig. 64: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 900°F.

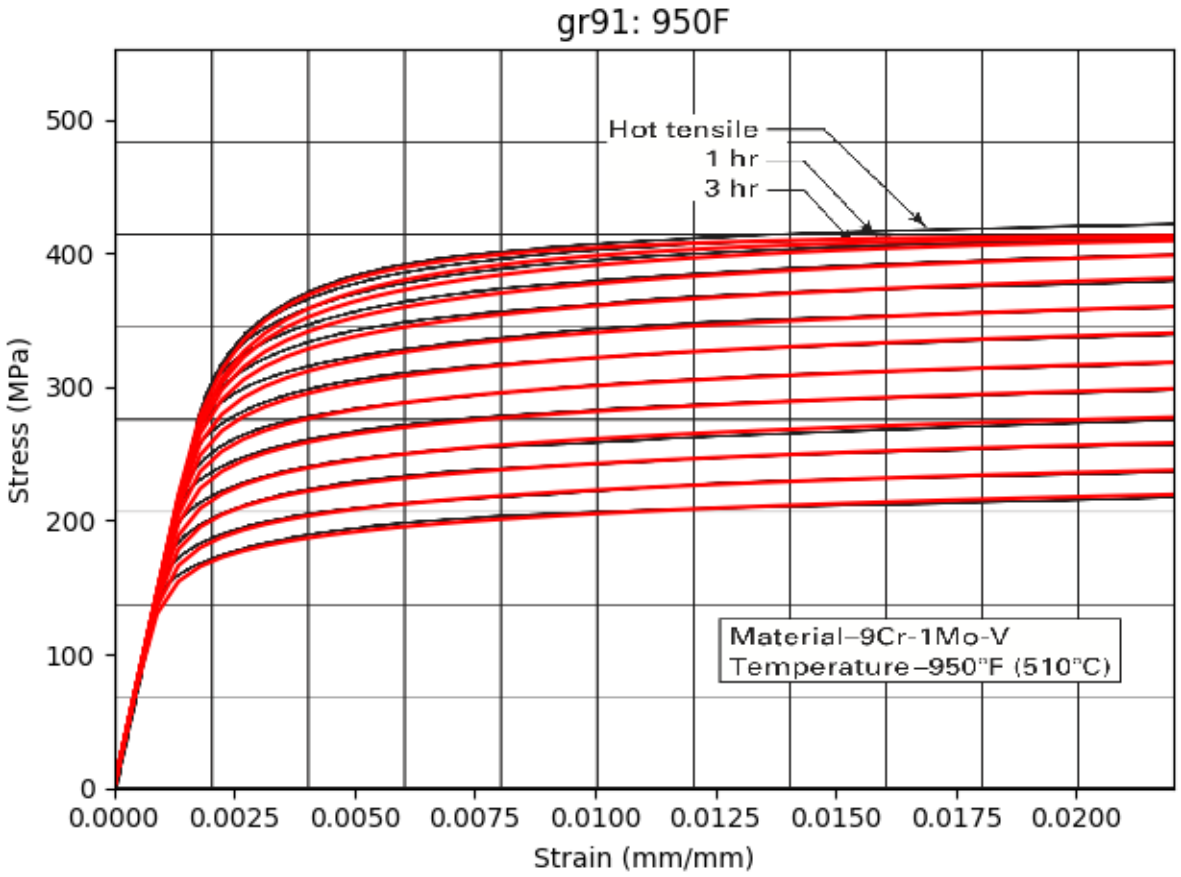

Fig. 65: Comparison between the current and implemented isochronous stress-strain curves for 9Cr-1Mo-V at 950°F.

PYTHON MODULE INDEX

h

INDEX